

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
PROGRAMŲ SISTEMŲ KATEDRA

Baigiamasis bakalauro darbas

**Turinio adaptavimas skirtingiems mobiliesiems įrenginiams**

Atliko: 4 kurso, 9 grupės studentas

Džiugas Baltrūnas

Darbo vadovas:

Saulius Maslinskas

Vilnius

2005

## Turinys

Santrumpų ir terminų žodynas .....	3
Įvadas .....	4
<b>1. Turinys mobiliajame telefone .....</b>	<b>5</b>
<b>1.1. Turinio formatai .....</b>	<b>5</b>
<b>1.2. Pagrindinės problemos .....</b>	<b>8</b>
<b>2. Mobiliojo ryšio telefono atpažinimas .....</b>	<b>9</b>
<b>2.1. Vartotojo naršyklės antraštė User-Agent .....</b>	<b>9</b>
<b>2.2. Vartotojo naršyklės profilis UAProf .....</b>	<b>10</b>
<b>2.3. Telefono modelio nustatymas pagal IMEI .....</b>	<b>10</b>
<b>3. Telefono savybių nustatymas .....</b>	<b>13</b>
<b>3.1. Accept* antraštės .....</b>	<b>13</b>
<b>3.2. Vartotojo naršyklės (angl. User Agent) profilis .....</b>	<b>14</b>
<b>3.3. Paieška savybių duomenų bazėje .....</b>	<b>15</b>
<b>4. Turinio adaptavimas .....</b>	<b>18</b>
<b>4.1. Paruoštas turinys skirtingoms telefonų modelių grupėms .....</b>	<b>18</b>
<b>4.2. Adaptavimas turinio serveryje .....</b>	<b>19</b>
<b>4.3. Adaptavimas tarpinėje grandyje .....</b>	<b>20</b>
<b>4.4. Adaptavimas vartotojo pusėje .....</b>	<b>21</b>
<b>5. Siūlomi sprendimai .....</b>	<b>23</b>
<b>5.1. Telefono modelio ir savybių atpažinimas naudojant IMEI ir WURFL .....</b>	<b>23</b>
<b>5.2. Telefono modelio ir jį atitinkančio User-Agent duomenų bazė .....</b>	<b>25</b>
<b>5.3. Turinio adaptavimo platforma .....</b>	<b>28</b>
<b>5.4. Atvirojo kodo įrankiai turinio transformacijai .....</b>	<b>37</b>
<b>5.5. WML/XHTML puslapiavimas .....</b>	<b>39</b>
<b>5.6. Galimos alternatyvos .....</b>	<b>41</b>
Išvados .....	42
Summary .....	43
Literatūros ir šaltinių sąrašas .....	44
<b>1 priedas. Nokia 6230 vartotojo naršyklės profilio UAProf fragmentas .....</b>	<b>45</b>
<b>2 priedas. WML puslapiavimui naudojamos <i>addElement</i> ir <i>getNodeSize</i> f-jos .....</b>	<b>47</b>

## **Santrumpų ir terminų žodynas**

MSISDN – Mobile Subscriber ISDN Number. Mobilaus ryšio vartotojo tel. nr.

SMS – Short Messaging Service. Trumųjų žinučių paslauga.

MMS – Multimedia Messaging Service. Daugialypės terpės žinutės.

GSM – Global System for Mobile Communications. Mobiliojo ryšio sistema.

MIDP – Mobile Information Device Profile. Mobiliosios Java redakcijos (J2ME) profilis.

CSCD – Connected Limited Device Configuration. J2EE konfigūracija.

HTTP – Hyper Text Transfer Protocol. Hiperteksto protokolas.

WAP – Wireless Application Protocol. Mobilusis protokolas.

IMEI – Internation Mobile Equipment Identifier. Mobilaus įrenginio identifikatorius.

MSC – Mobile Switching Center. Centrinė komutacinė stotis.

EIR – Equipment Identification Register. Mobilųjų įrenginių registras.

HLR – Home Location Register. Operatoriaus vartotojų registras.

TAC – Type Approval Code. IMEI kodo dalis, identifikuojanti telefono modelį.

SVN – Software Version Number. Programinės įrangos versija.

SRN – Serial Number. IMEI serijos numeris.

SS7 – Signaling System 7. Telefoninės signalizacijos sistema.

OMA – Open Mobile Alliance. Standartų organizacija.

STI – Standart Transcoding Interface. Standartizuotas perkodavimo interfeisas.

WURFL – Wireless Universal Resource File. Telefonų savybių duomenų bazė XML formatu.

AMR – Adaptive Multi-Rate. Garsinis formatas.

## Ivadas

Didėjantis skirtingų mobiliųjų įrenginių kiekis kelia vis didesnius iššūkius mobiliųjų turinio ir paslaugų tiekėjams bei mobiliojo ryšio operatoriams. Informacijos pateikimas eiliniam mobiliojo telefono naudotojui tampa vis sudėtingesnis dėl skirtingų telefonų charakteristikų. Turinio tiekėjai yra priversti kurti arba pirkti brangią programinę įrangą, kuri leistų pateikti paslaugas, nepriklausančias nuo vartotojo telefono aparato. Be to, būtina pastoviai sekti naujas telefonų savybes, tuo pačiu užtikrinant paslaugų kokybę senesnių telefonų naudotojams.

Vartotojui, naudojančiam mobilių telefoną, turi būti sudaryta galimybė naudotis skirtingomis paslaugomis bei peržiūrėti ar atsisiųsti skirtingus turinio tipus nepriklausomai nuo turimo telefono modelio. Šiai problemai spręsti yra būtini telefono modelio ir jo savybių nustatymo bei turinio adaptavimo mechanizmai.

Šio darbo tikslas ir yra aptarti pagrindinius informacijos turinio, kuriems aktuali turinio adaptavimo problema, tipus, išnagrinėti pagrindinius telefono modelio atpažinimo, jo savybių nustatymo bei turinio adaptavimo principus ir pateikti siūlomus projektinius sprendimus, besiremiančius aptariamais mechanizmais ir padedančius spręsti minimas problemas.

Atliekant darbą buvo susidurta su didele standartų, apibrėžiančių mobiliąsias paslaugas bei turinio formatus, gausa, todėl vienas iš šio darbo pagrindinių uždavinių buvo sprendimo, leidžiančio taikyti universalius turinio adaptavimo mechanizmus, paieška.

Darbo metu buvo konsultuojamasi su vienu iš Lietuvos mobiliojo ryšio operatorių, darbe taip pat panaudotos sukauptos žinios iš studijų metu atlikto grupinio projekto „Informacinės ir pramoginės WAP portalų platformos architektūra“ bei kursinio projekto „Turinio adaptavimas skirtingiems mobiliams įrenginiams“.

## 1. Turinys mobiliajame telefone

Pagrindinė mobiliojo ryšio telefono paskirtis – priimti ir gauti telefoninius balso skambučius. Beveik visi GSM ryšio telefonai taip pat palaiko ir trumpųjų pranešimų siuntimo bei gavimo sistemą SMS (angl. Short Messaging Service). Prieš keletą metų GSM standartą papildė paketinio duomenų perdavimo paslauga GPRS (angl. General Packet Radio Service), atvėrusia platesnes galimybes telefonų vartotojams naudotis mobiliuoju internetu. Populiariausias protokolas, veikiantis mobiliojo interneto pagrindu – WAP (angl. Wireless Application Protocol), leidžiantis telefonu naršyti internete bei siųsti ir gauti iliustruotas MMS (angl. Multimedia Messaging System) žinutes. Tiesa, pastaroji paslauga atsirado šiek tiek vėliau, nei pats WAP protokolas, tačiau yra suderinama su beveik visais per paskutinius keletą metų pagamintais mobiliaisiais GSM ryšio telefonais.

### 1.1. Turinio formatai

Kiekvienu iš minėtųjų informacijos kanalų – SMS, WAP ar MMS – galima perduoti tekstinę informaciją ar tam tikrą formato turinio objektą. Trumpai aptarsime turinio formatus, būdingus kiekvienam iš informacijos kanalų.

#### 1.1.1. SMS

Vienas SMS žinutės simbolis gali būti koduojamas 7, 8 arba 16 bitų. Maksimalus vienoje žinutėje perduotų duomenų kiekis – 140 baitų. Žinutės, kurių simboliai užkoduoti 7 ar 16 bitų, naudojamos tekstinei informacijai siųsti. Visus standartinės SMS koduotės, kurią numato GSM 03.38 standartas, simbolius galima užkoduoti 7 bitais, todėl į vieną SMS žinutę galima sutalpinti 160 tokių simbolių. Įvairūs nacionalinių raidynų simboliai, nepatenkantys į GSM 03.38 alfabetą, koduojami UCS2 (angl. Universal Character Set) koduote, kurios vienas simbolis užima 16 bitų informacijos. Tokias žinutes gali parodyti tik naujesni telefonų modeliai. 8 bitų, arba dvejetainės žinutės yra skirtos įvairiems turinio objektams arba sisteminio pobūdžio informacijai perduoti. 1 lentelėje pateikti pagrindiniai dvejetainėmis žinutėmis perduodami objektai, jų paskirtis bei suderinamumo problemos.

Turinio objektas	Paskirtis	Suderinamumas
Balso, fakso, el. pašto notifikacijos	Vartotojui sutikus, atidaroma gauta balso pašto žinutė, faksas ar elektroninis	Beveik visi Nokia modeliai

	laiškas	
Flash	Mirksintis arba įspėjamasis (angl. alert) SMS pranešimas	Visi Nokia telefonai, kai kurie Siemens, Ericsson, Motorola ir keletas kitų
Nespalvoti logotipai, atvirukai ir monofoninės melodijos	Operatoriaus logotipui, įeinančių skambučių melodijai pakeisti, pramogoms	Skirtingi standartai modelių grupėms (EMS, Nokia Smart Messaging, Siemens SEO)
WAP Push	Formuojamas pranešimas su URL (adresu internete). Vartotojui sutikus WAP naršyklėje yra atidaromas nurodytas URL adresas.	Dauguma Nokia, kai kurie Siemens modeliai
WAP OTA (Over The Air)	Automatiniai operatoriaus nustatymai telefono WAP naršyklei (šliuzas, namų puslapis ir pan.)	Dauguma telefonų be TCP/IP steko
WAP OTA OMA	Automatiniai operatoriaus nustatymai telefono tinklo prieigai ir MMS žinutėms	Dauguma telefonų su TCP/IP steku

1 lentelė. Pagrindiniai binarinių SMS formatai.

### 1.1.2.MMS

MMS žinutė siunčiama išnaudojant tiek SMS, tiek WAP technologijas. Pirmiausia į mobilų telefoną yra išsiunčiamas pranešimas apie gautą MMS, tada pats telefonas WAP pagalba atsisunčia MMS žinutę. Žinutė yra pateikiama prezentacijos (judančių kadru) pavidalu, aprašomu SMIL (angl. Synchronized Media Integration Language) kalba. MMS dydį riboja mobilusis įrenginys (pavyzdžiui, Nokia 3100 leidžia maksimalų 45 kilobaitų žinutės dydį). Persiunčiama informacija gali būti teksto, vaizdo ar garso failai, bandomos ir video failų galimybės. 2 lentelėje pateikti pagrindiniai MMS žinutės sudedamųjų failų formatai.

Tipas	Formatai	Suderinamumas
Garsas	AMR	Dauguma MMS palaikančių telefonų
	MP3	Naujausi Nokia, Siemens, Samsung modeliai
	SP-MIDI	Visi polifonines melodijas palaikantys telefonai, tačiau skiriasi maksimalus kanalų melodijoje kiekis ir kitos savybės
Paveikslėliai	GIF87a, GIF89a	Naujesni Nokia, kai kurie Motorola ir Siemens telefonai
	JPEG (su JFIF)	Dauguma telefonų, kuriuose įmontuotos fotokameros
	PNG	Labai nedaug telefonų
	WBMP	Visi telefonai, palaikantys WAP protokolą
Prezentacija	SMIL	Beveik visi MMS palaikantys telefonai, tačiau skiriasi įvairūs atributai skirtingoms telefonų grupėms
Video	3GPP MPEG 4	Tik naujausi telefonai (daugiausia Nokia)

2 lentelė. Pagrindiniai MMS žinutę sudarančių failų formatai.

### 1.1.3. WAP

Populiariausias WAP panaudojimas yra specialių interneto puslapių, konstruojamų WML (angl. Wireless Markup Language) kalba, naršymas. Vieno WML puslapio apimtį riboja telefono WAP naršyklės deko dydis, todėl tokiuose puslapiuose turi būti puslapiavimo mechanizmai, kuriuos aptarsime paskutiniame šio darbo skyriuje.

WAP naršyklės pagalba galima atsisiųsti beveik visus (išskyrus MMS prezentaciją) MMS žinutėse naudojamus turinio tipus bei kitus WAP protokolu siunčiamo turinio formatus. Svarbiausi iš jų yra pateikti 3 lentelėje.

<b>Turinio formatas</b>	<b>Paskirtis</b>	<b>Suderinamumas</b>
WML 1.1	Teksto ir grafinių elementų išdėstymas	Visi telefonai, palaikantys WAP 1.x, tačiau skiriasi maksimalus puslapio (kortelės) dydis
WML 2.0, XHTML	Teksto ir grafinių elementų išdėstymas	Visi telefonai, palaikantys WAP 2.0
WBMP	Nespalvoti paveikslėliai	Visi WAP palaikantys telefonai
GIF, PNG, JPEG	Spalvoti paveikslėliai	Dauguma Nokia, kai kurie Motorola ir Siemens modeliai
CSS	Puslapio elementų ir jų išsidėstymo stiliai	Tik WAP 2.0 standartą palaikantys telefonai
Kiti (JAVA ar Symbian programos bei žaidimai ir pan.)	Kiti turinio formatai, kurie gali būti atsisiunčiami WAP naršyklės pagalba	Priklauso nuo konkretaus telefono galimybių

3 lentelė. Pagrindiniai WAP protokolu siunčiamo turinio formatai.

## 1.2. Pagrindinės problemos

Problematiką geriausiai galėtų pailiustruoti pavyzdys – elektroninė parduotuvė, orientuota į mobiliojo ryšio vartotojus. Tarkime, kad prekių sąrašas ir užsakymas veikia WAP svetainėje, o įvairias naujienas parduotuvė vartotojams siunčia SMS ir MMS žinutėmis. Kaip sukurti WAP puslapį, kuris korektiškai būtų matomas įvairių telefonų naudotojams? Į kokias dalis suskaidyti vieną puslapį taip, kad vartotojai, kurių telefonų WAP naršyklė turi mažesnę deko dydį, vienodai gerai matytų? Kaip nusiųsti MMS žinutę su prekės iliustracija ir aprašymu, kad ją perskaitytų visi paslaugą užsisakę vartotojai? Kaip išpėti vartotojus apie naujai atsiradusią prekę WAP kataloge ir pateikti jiems nuorodą į WAP puslapį? Tai tik dalis problemų, kylančių dėl skirtingų telefonų modelių charakteristikų.

Minėtoms problemoms spręsti yra būtini vartotojo telefono modelio ir jo savybių atpažinimo bei turinio adaptavimo mechanizmai, kuriuos šiame darbe ir panagrinėsime.

## 2. Mobiliojo ryšio telefono atpažinimas

Tipinis informacijos gavimo modelis veikia principu užklausa (angl. request) – atsakymas (angl. response). Vadinasi, vartotojas, norėdamas gauti tam tikrą informaciją, savo mobiliuoju telefonu inicijuoja užklausa, skirtą paslaugų tiekėjui. Užklausa gali būti atliekama vienu iš šių kanalų: įprastu skambučiu, SMS žinute, telefono naršyklę nukreipiant į tam tikrą WAP puslapį arba siunčiant MMS žinutę. Kartais užklauskos iniciatorius yra ne pats vartotojas, o paslaugų tiekėjas, tuomet vartotojas yra identifikuojamas operatoriaus jam suteiktu telefono numeriu, dar vadinamu MSISDN (angl. Mobile Subscriber ISDN Number). Dažniausiai tokiu būdu yra teikiamos įvairios periodinės paslaugos – informacijos prenumerata, siunčiama reklaminė informacija ir pan. Kiekvienu iš minėtų atvejų norint pateikti vartotojui turinį, kuris gali skirtis priklausomai nuo telefono savybių, visų pirma reikia sužinoti vartotojo telefono modelį.

### 2.1. Vartotojo naršyklės antraštė User-Agent

Mobilaus telefono naršyklė kiekvienos užklauskos metu paprastai siunčia save identifikuojančią informaciją. Šis būdas yra vienas seniausiai ir plačiausiai taikomų, kadangi tokiu pačiu principu yra identifikuojamos įprastos interneto naršyklės bei jų versijos. User-Agent antraštėje paprastai yra perduodama informaciją apie naršyklę bei vartotojo operacinę sistemą, kartais pateikiama papildoma informacija apie aparatūrą. Šią antraštę numato HTTP (angl. Hyper Text Transfer Protocol) protokolas. Remiantis RFC 2616 [RFC2621], User-Agent antraštė turėtų būti sudaroma tokiu principu:

*User-Agent* = "User-Agent" ":" 1\*( *produktas* | *komentaras* )  
*produktas* = simbolių eilutė ["/" *produkto-versija*]  
*produkto-versija* = simbolių eilutė

Pateikiame pavyzdį, kaip atrodo mobiliojo telefono Nokia 6230 User-Agent antraštė:

*User-Agent: Nokia6230/2.0 (04.44) Profile/MIDP-2.0 Configuration/CLDC-1.1*

Šiuo atveju produktą atitinka Nokia6230, produkto versiją – 2.0 (04.44), o toliau yra pateikiama papildoma informacija – apibrėžiamas telefono CLDC (angl. Connection Limited Device Configuration) – tai J2ME (angl. Java 2 Platform, Micro Edition) konfigūracija, skirta

nešiojamiems įrenginiams, dažniausiai maitinamiems iš baterijos ir pasižymintiems nedideliu atminties dydžiu bei procesoriaus galingumu. Kol kas vienintelė CLDC konfigūracija yra MIDP (angl. Mobile Information Device Profile) profilis.

## 2.2. Vartotojo naršyklės profilis UAProf

Be User-Agent antraštės daugelis naujesnių mobiliųjų telefono naršyklių taip pat siunčia antraštę su nuoroda į User Agent profilį (UAProf arba x-wap-profile) – XML formato dokumentą, kuris identifikuoja tiek telefono modelį, tiek jo savybes. Telefono Nokia 6230 atveju ši antraštė atrodo štai taip:

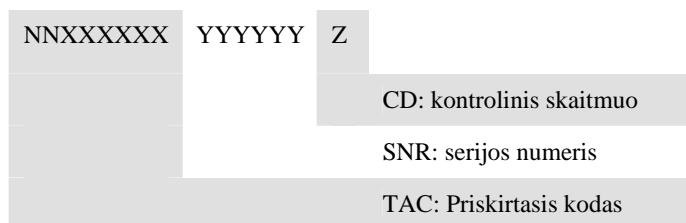
*x-wap-profile: "http://nds1.nds.nokia.com/uaprof/N6230r200.xml"*

Ši antraštė nusako tiek patį telefono modelį, tiek jo savybes. User Agent profilio struktūrą aptarsime kitame šio darbo skyriuje, o fragmentas iš Nokia 6230 profilio pateiktas šio darbo pirmame priede.

## 2.3. Telefono modelio nustatymas pagal IMEI

Kiekvienas mobilus GSM ryšio telefonas turi unikalų kodą, identifikuojantį telefoną. Tai tarptautinis mobilus įrenginio identifikatorius IMEI (angl. International Mobile Equipment Identifier), sudarytas iš 15 skaitmenų ir vienareikšmiškai identifikuojantis konkretų bevielį telefoną ar kitą komunikacinį įrenginį, dirbantį GSM tinkle.

Struktūra, kokios dalys nuo 2004 m. balandžio 1 dienos sudaro IMEI kodą, pateikta 4 lentelėje.



4 lentelė. IMEI struktūra nuo 2004 04 01.

Pirmieji du TAC (angl. Type Allocation Code) kodo (NNXXXXXX) skaitmenys NN identifikuoja kodą priskyrusią instituciją (angl. Reporting Body), o like 6 XXXXXX – tipo identifikatorius, priskirtas minėtos institucijos. SNR (angl. Serial Number) – tai unikalus 6

skaitmenų serijos numeris, paskirtas konkrečiam skyriui. CD (angl. Check Digit) – kontrolinis skaitmuo, priklausantis nuo pirmųjų 14 skaitmenų. IMEI informacijos su operatorimi apsikeitimo metu šis skaičius nėra persiunčiamas. Kai kurie telefonai papildomai siunčia ir programinės įrangos versijos numerį SVN (angl. Software Version Number) – dviženklį skaičių, nepriklausantį nuo IMEI, bet jį papildantį.

Dažniausiai IMEI numeris yra užrašomas ant priešingos displejui telefono pusės, jį taip pat galima sužinoti telefone surinkus kombinaciją *\*#06#*. Kai telefonas yra įjungiamas, šis numeris yra siunčiamas ir tikrinamas mobiliųjų įrenginių registro EIR (angl. Equipment Identity Register) duomenų bazėje. Operatoriaus EIR skirstomas į tris registrus – Baltąjį (angl. white lists), pilkąjį (angl. grey lists) ir juodąjį (angl. black lists) sąrašus. Šių sąrašų panaudojimas priklauso nuo kiekvieno operatoriaus poreikių, tačiau tradiciškai baltąjį sąrašą sudaro tų telefonų IMEI numeriai, kuriems yra leidžiama naudotis tinklo paslaugomis, o juodajame sąrašė saugomi tie IMEI, kuriems yra uždrausta naudotis tinklo paslaugomis. Dažniausiai tai būna vogtų telefonų IMEI numeriai. Be baltojo ir juodųjų sąrašų tinklo administratoriai tam tikrus IMEI numerius gali įtraukti į pilkąjį sąrašą. Telefono aparatams, kurių IMEI yra šiame sąrašė, nėra draudžiama naudotis tinklu, tačiau dažniausiai šiame sąrašė esantys IMEI yra stebimi dėl vienokių ar kitokių priežasčių.

Toliau pagrindinis uždavinys yra telefono gamintojo bei modelio gavimas iš IMEI numerio. Kaip jau buvo minėta, TAC yra 8 skaitmenų tipo, arba kitaip, telefono gamintojo ir modelio identifikatorius. Svarbu paminėti, kad iki 2004 m. balandžio 1 d. TAC užkoduoti buvo skirti tik pirmieji 6 skaitmenys, o likę du turėjo būti 00 (iki 2003 m. sausio 1 d. jie reiškė galutinio surinkimo kodą FAC). Iš esmės TAC buvo išplėstas tam, kad ateityje nepritrūktų skaitmenų naujiems telefonų modeliams, todėl šiuo metu realiai užtenka pirmųjų 6 IMEI (o kartu ir TAC) skaitmenų, norint identifikuoti telefono modelį. Telefono Nokia 6230, pagaminto 2004 03 17, IMEI numerio struktūra pateikta 5 lentelėje.

352953	00	498774	3	
				SN
				FAC
				SNR
				TAC

5 lentelė. Nokia 6230 IMEI struktūra.

Tam, kad galėtume pagal TAC nusakyti telefono gamintoją bei modelį, reikalinga duomenų bazė, sudaryta iš TAC bei telefono modelio porų. Internete pavyko surasti ir viešą tokios informacijos duomenų bazę [NOBBI-TAC], kur jau sukaupta per 5000 skirtingų TAC serijų, o papildyti duomenų bazę gali kiekvienas norintis.

Šiame skyriuje aptarėme tris telefono modelio nustatymo mechanizmus – pagal User-Agent ir UAprof antraštes bei pagal IMEI TAC kodą. Kitame šio darbo skyriuje panagrinėsime, kaip, žinant telefono modelį, galima nusakyti jo savybes.

### 3. Telefono savybių nustatymas

Tam, kad konkretaus turinio objektą galėtume pritaikyti vartotojo telefonui, vien jo modelio žinoti nepakanka. Reikia išsiaiškinti, kokius turinio formatus žinomas telefonas gali apdoroti, kuris turinio formatas telefonui yra „priimtinesnis“, kokiomis kalbomis ar koduotėmis telefonas gali rodyti tekstą, kokia telefono displejaus spalvų raiška ir daugelį kitų savybių.

Tam, kad galėtume nustatyti telefono savybes, dažniausia reikia žinoti pati telefono modelį. Kaip jau minėjome, tai galima padaryti vartotojo naršyklės arba jos profilio antraščių pagalba arba identifikuojant telefono aparatą pagal IMEI numerį.

#### 3.1. Accept\* antraštės

Procesas, kuomet yra apsprendžiama, koks turinys iš kelių turimų turinio versijų yra tinkamiausias įrenginiui, vadinamas turinio „derybomis“ (angl. content negotiation). HTTP protokolas, naudojamas tiek įprastiems interneto, tiek WAP puslapiams parsisiųsti, numato turinio „derybų“ mechanizmą, kuomet naršyklė nurodo, kokio tipo informacija jai yra priimtina, o serveris, kuris pristato turinį, nusprendžia, kokią informaciją gražinti remiantis iš naršyklės gautais duomenimis.

Turinio „derybų“ metu žiniatinklo (angl. Web) serveris gali pats parinkti tinkamiausią objekto versiją, remdamasis naršyklės pateiktais duomenimis apie turinio tipą, kalbą bei koduotę. *Accept* antraštė užklausoje nurodo, kokius turinio tipus naršyklė gali priimti. *Accept-Charset* antraštės pagalba galima pasakyti, kokie simbolių rinkiniai yra priimtini naršyklei. Panašią paskirtį turi ir *Accept-Encoding* antraštė, kuri nurodo, kokiais turinio kodavimo (angl. content encoding) algoritmais gali būti siunčiamas atsakymas į užklausą. Galiausiai *Accept-Language* antraštė apibrėžia natūraliąsias kalbas, priimtinas turiniui.

*Accept* antraštėse gali būti pateikiami ne tik formatai, kuoduotės ar kodavimo algoritmai, bet ir jų svoriai. Tai atliekama prie užklausoje pridendant pirmumo faktorių (angl. preference factor)  $q$ , kur  $q$  gali būti reikšmė iš intervalo  $[0..1]$ . Jei šis faktorius nenurodomas, laikoma, kad jis lygus vienetui ir turi aukščiausią prioritetą. Pavyzdžiui, jei naršyklei UTF-8 koduotė yra priimtinesnė nei ISO-8859-1, tuomet antraštė galėtų siųsti tokį *Accept-Charset* koduočių sąrašą: utf-8; q=0.9, iso-8859-1; q=0.6. Žvaigždutės (\*) simbolis antraščių reikšmėse yra naudojamas turinio tipų grupavimui arba jo pagalba galima nusakyti, kad priimtinos yra visi simbolių rinkiniai, turinio koduotės ar kalbos [Son04].

Štai kokias Accept antraštes siunčia telefono Nokia 6230 WAP naršyklė:

*Accept: application/vnd.wap.wmlscriptc, text/vnd.wap.wml,  
application/vnd.wap.xhtml+xml, application/xhtml+xml, text/html, multipart/mixed, \*/\**  
*Accept-Charset: ISO-8859-1, US-ASCII, UTF-8; q=0.8, ISO-10646-UCS-2; q=0.6*  
*Accept-Encoding: gzip, deflate*  
*Accept-Language: lt-LT*

Matome, kad Accept antraštėje yra išvardinta tik keletas turinio tipų, kurie yra „priimtinausi“, o galiausiai nurodoma, kad telefonas „priims“ bet kokius turinio tipus. Svarbu paminėti, kad ši problema atsiranda dėl vis „protingesnių“ telefonų, kurie palaiko ne vieną dešimtį turinio tipų ir todėl visus juos siųsti kiekvienos užklaustos metu būtų neefektyvu. Įdomu, tačiau to paties Nokia 6230 telefono senesnės versijos Accept antraštėje įtraukia pilną turinio tipų sąrašą.

Nors HTTP turinio „derybos“ yra paprastas ir patogus būdas padaryti tam tikrus sprendimus apie telefono savybes, tačiau jis buvo sukurtas tradicinėms naršyklėms (Internet Explorer, Mozilla ir k.t.) ir todėl perneša tik ribotą kiekį informacijos apie įrenginį ir jo savybes.

### **3.2. Vartotojo naršyklės (angl. User Agent) profilis**

Vis didėjančiam telefonų su mobiliuoju internetu savybėms nusakyti nebepakanka ankščiau minėtų būdų telefono savybėms nusakyti. Žiniatinklo konsorciumas W3C (angl. World Wide Web Consortium) pasiūlė galimybių ir savybių profilių rinkinį CC/PP (angl. Composite Capabilities / Preferences Profile) – standartą, leidžiantį nusakyti įrenginių konfigūracijos parametrus bei galimybes. CC/PP galima vadinti karkasu, kuris apibrėžia priemonės skirtingiems įrenginiams nusakyti įvairias įrenginio savybes (displėjaus dydį, garso parametrus, duomenų perdavimo spartą ir pan.) vieninga forma. CC/PP yra paremtas resursų aprašymo karkasu RDF (angl. Resource Description Framework) [RDF]. Pats CC/PP savaime neapibrėžia aprašymo leksikono – yra pateikiamas tik leksikonų konstravimo būdas.

Atvirojo mobiliojo alianso OMA (buves WAP Forumas) sukurtas vartotojo naršyklės profilis UAProf (angl. User Agent Profile) standartas – tai konkretus CC/PP leksikonas, skirtas mobiliojo ryšio telefono aprašymui ir apibrėžia efektyvų būdą CC/PP aprašų perdavimui bevieliais tinklais. Mobilieji telefonai, tenkinantys UAProf specifikacijas, pateikia turinio tiekėjų serveriams savo galimybių aprašus. Turinio serveriai, šliuzai (angl. gateways)

ar tarpinės grandys (angl. proxies) taip pat gali pasinaudoti šia informacija adaptuojant turinį, kuris tiktų įrenginiui ir patenkintų vartotojo lūkesčius. Ši informacija yra pateikiama XML formatu ir padengia tokius įrenginio atributus:

- Aparatūrinė platforma – displejaus dydis, garsų grojimo galimybė, spalvų kiekis ir pan.
- Programinė platforma – operacinė sistema, turinio tipai, koduotės, siunčiamų duomenų kodavimas, garso ir vaizdo kodavimo standartai
- Tinklo charakteristikos – GSM/GPRS galimybė, apsaugos standartai, bevielio ryšio prieigos galimybės (Bluetooth, Infrared)
- Naršyklės charakteristikos – naršyklės pavadinimas ir versija, palaikomi turinio atvaizdavimo formatai (WML/XHTML), turinio tipai, skriptavimo kalbos, lentelių ar rėmelių puslapyje palaikymas
- WAP charakteristikos – WAP/WML palaikymas, deko dydis
- Push charakteristikos – push turinio tipai, push žinutės dydis

UAProf failai yra pakankamai dydeli – kuo daugiau savybių telefonas pasižymi, tuo failas didesnis. Pavyzdžiui, telefono Nokia 6230 UAProf failas užima apie 17 kilobaitų (fragmentas pateiktas pirmame priede). Būtent dėl šios priežasties standartas numato, kad serveriui bus perduodama tik nuoroda (URL) į įrenginio profilį. Tai atliekama minėtos *x-wap-profile* HTTP antraštės pagalba:

*x-wap-profile: "http://nds1.nds.nokia.com/uaprof/N6230r200.xml"*

Tai yra nuoroda serveriui, kur yra saugomas profilio failas. Turinio serveris, gavęs šią nuorodą, ją parsisiunčia iš įrenginių profilių saugyklos ir failą gali išsaugoti savo duomenų bazėje siekiant išvengti pakartotinių atsisuntimų. Svarbu paminėti, kad tiek WAP šliuzas (angl. WAP gateway), tiek HTTP tarpinė grandis (angl. proxy) privalo persiųsti UAProf antraštės kiekvienos užklauskos metu.

### **3.3. Paieška savybių duomenų bazėje**

UAProf profilis turi keletą esminių trūkumų. Visų pirma, kadangi standarto autorius yra OMA aljansas, nėra jokių garantijų, kad pats standartas nepasikeis (taip yra buvę praktikoje).

Taip pat nėra jokių garantijų, kad informacija, esanti UAProf profilyje, yra patikima. Taip gali atsitikti tuomet, kai telefonų gamintojai naujam telefono modeliui tiesiog priskiria profilį, skirtą panašiam, bet senesniai telefonui, arba pačiame UAProf XML dokumente įviliama klaida. Svarbu pastebėti ir tai, kad kai kurie telefonai apskritai nesiunčia nuorodos į vartotojo naršyklės profilį UAProf – tai būdinga Lietuvoje plačiai naudojamiems kai kuriems Motorola telefonų modeliams, pavyzdžiui C-450.

Minėta problema galėtų būti išspręsta sudarant nepriklausomą telefonų savybių duomenų bazę. Kadangi dėl augančio telefonų skaičiaus ir naujų savybių tokios duomenų bazės administravimas yra ypač komplikotas, vienas iš sprendimų galėtų būti jau egzistuojančios duomenų bazės panaudojimas. Kaip tik tokią duomenų bazę siūlo atvirojo kodo (angl. Open Source) projektas WURLF (angl. Wireless Universal Resource File). WURLF [WURLF] galima apibrėžti kaip XML dokumentą, kuris apima daugelį rinkoje esančių mobilių telefonų savybių. Kadangi WURLF duomenų bazė yra sudaroma ir atnaujinama savanorių iš viso pasaulio, ji tampa nepriklausoma nuo UAProf profiliuose esančių klaidų. Taip pat svarbu paminėti, kad įdiegus WURLF, nebereikia kiekvienos užklauskos metu atsisiųsti telefono UAProf profilio, taip sutaupant nemažai duomenų perdavimo kaštų.

WURLF XML failas yra sudaromas remiantis principu, kad vieno gamintojo naujesni telefono modeliai arba to paties modelio naujesnės versijos paveldi didžiąją dalį senesnio modelio savybių. Taip yra išvengiama duomenų dubliavimo, palengvina duomenų atnaujinimą bei sumažina telefono modelių ir jų savybių matricos dimensijas. Pavyzdžiui, fragmentas iš telefono Nokia 6230 „bazinės“ versijos aprašo atrodo taip:

```
<device user_agent="Nokia6230" actual_device_root="true" fall_back="nokia_generic_series40_dp20"
  id="nokia_6230_ver1">
  <group id="product_info">
    <capability name="model_name" value="6230" />
  </group>
  <group id="storage">
    <capability name="max_deck_size" value="102400" />
    <capability name="max_no_of_bookmarks" value="50" />
  </group>
  <!-- ... -->
</device>
```

Visos kitos telefono Nokia 6230 versijos aprašomos tik pateikiant nuorodą į „bazinę“ versiją. Tai nurodo *device* atributas *fall\_back*:

```
<device user_agent="Nokia6230/2.0 (03.04) Profile/MIDP-2.0 Configuration/CLDC-1.1"
  fall_back="nokia_6230_ver1" id="nokia_6230_ver1_sub0304" />
```

```
<device user_agent="Nokia6230/2.0 (03.06) Profile/MIDP-2.0 Configuration/CLDC-1.1"  
fall_back="nokia_6230_ver1" id="nokia_6230_ver1_sub0306" />  
<device user_agent="Nokia6230/2.0 (03.12) Profile/MIDP-2.0 Configuration/CLDC-1.1"  
fall_back="nokia_6230_ver1" id="nokia_6230_ver1_sub0312" />  
<!-- ... -->
```

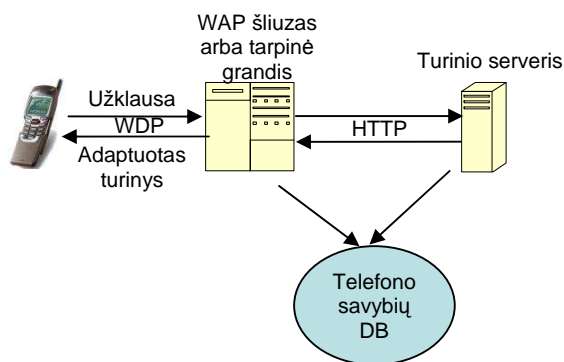
Matome, kad ir „bazinė“ Nokia 6230 paveldi 40 serijos Nokia telefonų savybes. Tokiu būdu sukuriamas hierarchinis medis, kurio vaikuose aprašomos tik tos savybės, kurių nėra arba kurios skiriasi nuo tėvinės viršūnės.

Svarbu pastebėti, kad ir aptarta WURLF telefonų savybių duomenų bazė turi tam tikrų trukumų. Visų pirma, joje sukaupta tik tam tikros dalies telefonų modelių savybės, o kadangi ši duomenų bazė yra pildoma savanorių, joje taip pat galimos klaidos ar netikslumai.

Šiame skyriuje aptarėme, kaip žinant telefono modelį galima nusakyti vartotojo naršyklės profilio UAProf pagalba arba atliekant paiešką telefonų savybių duomenų bazėje. Bendru atveju, t.y., kai telefono modelis nebūtinai yra žinomas, paprasčiausioms telefono savybėms nusakyti tinka ir HTTP *Accept* šeimos antraštės. Kitame šio darbo skyriuje panagrinėsime, kokie yra pagrindiniai turinio adaptavimo mobiliesiems įrenginiams mechanizmai.

#### 4. Turinio adaptavimas

Turinio adaptavimas – tai procesas, kuomet vartotojui, priklausomai nuo informacijos, apibūdinančios konkrečią situaciją, yra pateikiamas iš originalaus turinio objekto dinamiškai sugeneruotas arba iš kelių to paties turinio versijų parinktas objektas, atitinkantis vartotojo poreikius ir telefono galimybes. Paprastai adaptavimo metu vienas turinio objekto formatas yra transformuojamas į kitą arba pakeičiamos tam tikros objekto savybės – sumažinama paveikslėlio raiška paveikslėlyje ar kanalų skaičius polifoninėje melodijoje ir pan. Priklausomai nuo situacijos ir poreikių, turinio adaptavimą gali atlikti pats mobilusis telefonas, jis taip pat gali būti atliekamas tarpinėje grandyje (angl. proxy) arba turinio serveryje. Bendru atveju tokio adaptavimo schema pavaizduota 1 pav.



1 pav. Bendra turinio adaptavimo schema.

Kita alternatyva yra išvengti adaptavimo, paruošiant to paties turinio versijas skirtingoms telefonų grupėms. Šiame skyriuje ir pakalbėsime apie kiekvieną iš minėtų turinio adaptavimo mechanizmų.

##### 4.1. Paruoštas turinys skirtingoms telefonų modelių grupėms

Vienas iš būdų supaprastinti turinio adaptavimo procesą yra kelių turinio versijų paruošimas. Tokiu atveju pagrindinis uždavinys yra parinkti telefono modeliui labiausiai tinkamą turinio versiją. Jeigu adaptavimo metu nerandama turinio versijos, tiksliai tinkančios telefono modeliui, tuomet galima taikyti transformavimo mechanizmus tai versijai, kuri yra „artimiausia“, šitaip apsisaugant nuo didesnio duomenų praradimo ar iškreipimo transformacijos metu.

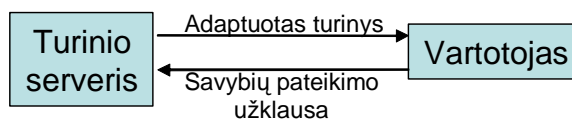
Nors šis būdas ir supaprastina turinio adaptavimo procesą, tačiau reikalauja daug laiko, specifinės programinės įrangos ir kartais specifinių žinių iš turinio tiekėjų, kadangi kiekvieną

naujai į duomenų bazę įkeliamą turinio objektą reikia konvertuoti į formatus, suprantamus atskiroms telefonų grupėms. Kita problema iškyla ir su turinio autorystės teisėmis, mat tampa neaišku, ar tam tikras savo savybes pakeitęs objektas nėra iškraipytas lyginant su originalia turinio versija.

Vis dėl to, lyginant su dinaminio transformavimu, atliekamu automatiškai, šis būdas turi savų privalumų ir yra neišvengiamas dėl minėtų iškraipymų galimybės bei dviprasmiškų situacijų, kurios galėtų iškilti atliekant dinaminį transformavimą.

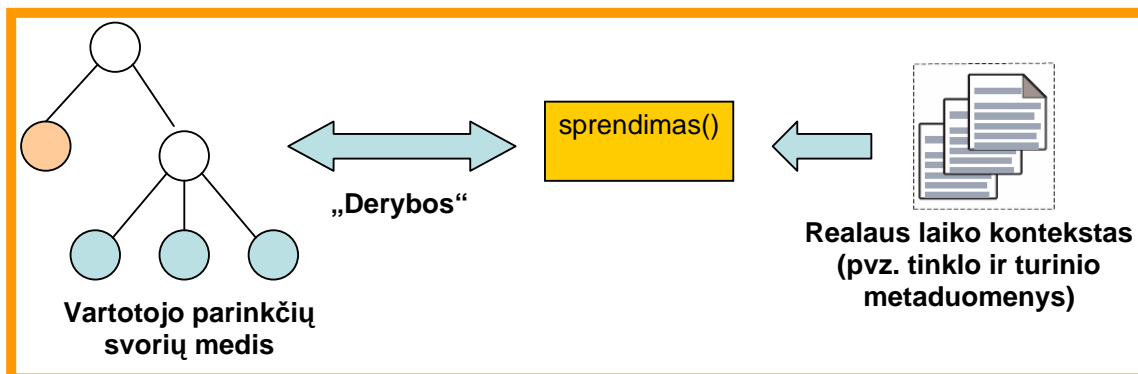
#### 4.2. Adaptavimas turinio serveryje

Jau aptartų kelių turinio versijų paruošimo problemas iš dalies sprendžia dinaminis turinio adaptavimas, atliekamas turinio tiekėjo serveryje (2 pav.).



2 pav. Turinio adaptavimas serveryje.

Siekiant minimizuoti informacijos praradimo ar jos iškraipymo tikimybę, prieš taikant adaptavimo ir transformacijos mechanizmus, būtina žinoti ne tik vartotojo telefono modelio savybes, bet ir tam tikras vartotojo parinktis. Pastaroji informacija dar yra vadinama kontekstine, o pats adaptavimas – vartotojiškas (angl. user-centric) [LL02]. Svarbiausia tokio adaptavimo dalis yra sprendimų variklis, kuris, naudodamasis kontekstine informacija apsprendžia, kuri turinio versija iš turimų yra tinkamiausia konkrečiai situacijai. Kontekstinę informaciją sudaro vartotojo parinkčių profilis (vartotojui tinkama spalva, objektų proporcijos, modalumo ir užlaikymo leidžiamos normos ir pan.), šiame darbe jau minėtas įrenginio profilis bei tinklo charakteristikos (duomenų perdavimo sparta). Šią informaciją papildo turinio objektų meta duomenys (dimensijos, paskirtis, įmanomos spalvų raiškos). Surinkus tokią informaciją sprendimų variklis atlieka turinio „derybų“ procesą [LL02], schematiškai pavaizduotą 3 pav.



3 pav. Turinio „derybų“ procesas.

Vartotojo parinkčių svorių medis yra sudaromas automatiškai, remiantis minėtomis vartotojo parinktimis. Kiekvieną medžio viršūnę atitinka turinio versija. Kuo labiau ji tenkina vartotojo pasirinkimus, tuo didesnis svoris jai priskiriamas. Panaudojant realaus laiko duomenis (tinklo ir turinio meta duomenis) funkcija `sprendimas()` nusprendžia, kuri turinio versija yra tinkamiausia:

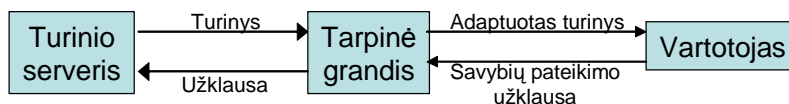
$$T / F = \text{sprendimas}(\text{svorių medžio šaka, turinio meta duomenys, tinklo charakteristika, įrenginio savybės})$$

Ši funkcija veikia iteratyviai, nagrinėdama kiekvieną svorių medžio šaką tol, kol gaunama T (True) reikšmė. Tai reiškia, kad gauta turinio versija yra tinkamiausia ir vartotojo telefonas ją galės korektiškai atvaizduoti. Jeigu tokia turinio versija nebuvo iš anksto paruošta, tuomet kviečiamos transformavimo funkcijos, verčiančios vieną turinio versiją į kitą. Vartotojo parinkčių svorių medis apeinamas vienu iš paieškos medyje algoritmų (SSL, ORST, NORTST, SSL-NORST).

Egzistuoja ir paprastesnių turinio adaptavimo serveryje mechanizmų, atliekančių elementarius failų formatų konvertavimus. Pateiktu pavyzdžiu norėta parodyti, kad atliekant adaptavimo procesus, svarbu atsižvelgti ne tik į vartotojo telefono savybes ar tinklo charakteristikas, bet ir į tai, koks turinys pačiam vartotojui yra priimtinesnis. Štai čia ir yra viena didžiausių šio mechanizmo problemų – kaip tokius duomenis surinkti?

### 4.3. Adaptavimas tarpinėje grandyje

Tarpinėje, arba nepriklausomoje grandyje taip pat gali būti atliekami adaptavimo mechanizmai (4 pav.).

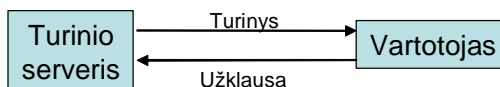


4 pav. Adaptavimas tarpinėje grandyje.

Tarpinėje grandyje atliekami adaptavimo mechanizmai yra gana riboti. Taip yra todėl, kad priešingai nei dviem aukščiau minėtais atvejais, tarpinė grandis gali operuoti tik originalia turinio versija bei neturi su turinio objektu susietos meta informacijos. Pavyzdžiui, tarpinė grandis galėtų atlikti HTML puslapių transformavimą į WML, tačiau tai gali pasisekti ne visada korektiškai, kadangi dėl riboto WAP deko dydžio WML puslapis yra pakankamai mažas, be to, kiekvienam telefonui jis yra skirtingas, todėl automatizuotu keliu tampa sudėtinga nuspręsti, kokiomis dalimis atlikti fragmentaciją.

#### 4.4. Adaptavimas vartotojo pusėje

Paskutinis būdas, kurį aptarsime – kai adaptavimą ir reikiamas transformacijas atlieka pats vartotojo telefono aparatas (5 pav.).



5 pav. Adaptavimas vartotojo pusėje.

Šis būdas nėra plačiai paplitęs, kadangi tik nedaugelis telefonų gamintojų yra įgyvendinę tokius mechanizmus. Nepaisant to, adaptavimas telefono aparate turi nemažai privalumų. Visų pirma, adaptavimu nebereikia rūpintis tarpinėms grandims ar turinio serveriams, spręsti šiame darbe jau minėtų su vartotojo naršyklės profiliu susijusių problemų – telefonas šias savybes „žino“ tiksliai ir iš anksto. Taip pat išsprendžiama problema, jei vartotojas, atsisuntęs turinio objektą, jį nori ne tik išsisaugoti, bet ir kam nors persiųsti. Taikant adaptavimo mechanizmus tarpinėje grandyje ar turinio serveryje, tai dažniausiai yra neįmanoma, kadangi turinio objektas yra pritaikomas tik tam telefonui, kuris pateikė užklausą.

Pagrindinė problema, susijusi su adaptavimu mobiliajame įrenginyje, yra turinio meta duomenų pateikimas telefonui. Turėdamas keletą lygiaverčio turinio objektų meta duomenis telefonas galėtų pats priimti sprendimą, kuri turinio versija jam yra tinkamiausia. Tokią galimybę iš dalies numato XHTML 2.0 standartas, kuris *object* elemente leidžia aprašyti

vieną ir tą patį objektą, tačiau skirtingais turinio tipais. Žinomesnis pavyzdys yra CSS (angl. Cascading Style Sheets) stilių su skirtingais *media* atributais naudojimas.

Kita problema yra didėjanti turinio formatų įvairovė. Daugelis formatų yra nereglamentuoti standartais, naudojami tik tam tikrų gamintojų, todėl vieno gamintojo telefono modeliui dažnai būna visai „nesuprantami“ kito gamintojo turinio formatai.

Vis dėlto, realesnių adaptavimo galimybių mobiliuosiuose įrenginiuose dar teks kiek palaukti, kadangi šis procesas reikalauja nemažai mikroprocesoriaus resursų bei atminties.

Šiame skyriuje išnagrinėjome tris turinio adaptavimo būdus, atliekamus turinio serveryje, tarpinėje grandinė arba pačiame mobiliajame įrenginyje. Atlikus visų šių būdų analizę paaikškėjo, kad kiekvienas jų turi tam tikrų privalumų bei trūkumų. Kitame šio darbo skyriuje aptarsime, kokie galimi universalūs sprendimai turinio adaptavimui.

## 5. Siūlomi sprendimai

Išnagrinėjus pagrindinius telefono modelio atpažinimo būdus, telefono savybių nustatymo galimybės bei pagrindinius turinio adaptavimo mechanizmų esminiai principus, šiame skyriuje aptarsime keletą siūlomų sprendimų tipinėms problemos spręsti. Visų pirma, panagrinėsime telefono modelio ir jo savybių atpažinimo IMEI ir WURFL pagalba architektūrą, taip pat išnagrinėsime standartizuotą perkodavimo interfeisą bei pateiksime galimą WML puslapiavimo projektą.

### 5.1. Telefono modelio ir savybių atpažinimas naudojant IMEI ir WURFL

Jei prieš pateikiant mobiliam telefonui skirtą turinį užklauso iniciatorius yra ne vartotojas, o paslaugų tiekėjas, arba jei informacijos perdavimo kanalas yra SMS žinutė, norint taikyti turinio adaptavimo mechanizmus, reikia iš anksto žinoti telefono, kuriam bus skirta informacija, modelį. Tokios duomenų bazės, saugančios telefono numerio (MSISDN) ir telefono gamintojo bei modelio porą, administratoriumi gali būti pats paslaugų tiekėjas, tačiau tuomet būtų neaišku, koku telefonu naudojasi vartotojas, jei jam paslauga yra teikiama pirmą kartą. Taip pat tokiai duomenų bazei yra reikalingas periodinis duomenų atnaujinimas, siekiant užtikrinti, kad informacija apie konkretaus vartotojo telefono modelį atitinka tikrovę (vartotojas gali būti pakeitęs telefoną arba į jį laikinai galėjo būti įdėta kita SIM kortelė).

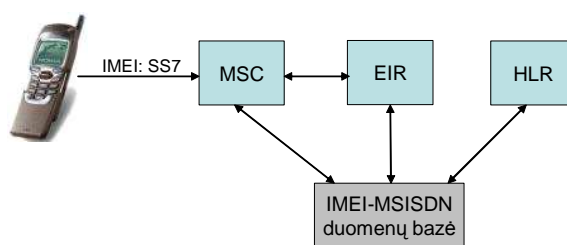
#### 5.1.1. IMEI perdavimas ir apklausimas per EIR

Mobilaus ryšio operatoriaus komutacinei stočiai MSC (angl. Mobile Switching Center) signalizacijos signalu SS7 (angl. Signaling System 7) IMEI numeris yra perduodamas ne tik tada, kai telefonas yra įjungiamas, bet ir kiekvieno skambučio, siunčiamos SMS žinutės ar duomenų perdavimo metu. MSC šią informaciją toliau perduoda tinklo įrenginių registru EIR bei tam tikrą laiką IMSI ir IMEI porą saugo savo atminties buferiuose. Tokiu būdu operatorius turi galimybę periodiškai užklausti EIR tokių duomenų ir taip pildyti IMSI ir IMEI porų duomenų bazę.

Teoriškai IMEI taip pat galima persiųsti iš MSC/VLR į HLR (MAP *UpdateLocation* užklauso metu) ir saugoti kaip vartotojo duomenis. Jei IMEI būtų saugomas HLR vartotojo profilyje, tai automatiškai padidintų vidutinį vartotojo profilio duomenų dydį bei atitinkamai sumažintų maksimalų vartotojų, kurių profilius gali saugoti HLR, skaičių. Šis padidėjimas yra palyginti nereikšmingas, kadangi IMEI saugojimui reikalingi 8 baitai bei lauko

identifikatorius [TSGS#19]. Vis dėlto, daugelyje GSM sistemų toks HLR išplėtimas nėra numatytas ir todėl jo įgyvendinimas bei testavimas gali kainuoti ypač brangiai.

Nustatyti vartotojo MSISDN žinant jo IMSI visada galima HLR pagalba. Tačiau pastovus HLR apklausinėjimas gali neigiamai atsiliiepti bendram operatoriaus sistemų funkcionavimui, kadangi HLR registras yra gyvybiškai svarbus įrenginys tinklo infrastruktūroje ir papildomas jo apkrovimas gali sulėtinti kitų sistemų darbą, todėl efektyvesnis būdas sužinoti MSISDN žinant IMSI yra apklausti kitas išorines operatoriaus sistemas, pavyzdžiui, apmokestinimo duomenų bazę, kuri taip pat saugo visų vartotojų MSISDN bei IMSI poras.



6 pav. IMEI apklausimas

Siūlomas sprendimas kaip tik ir yra nepriklausomos sistemos, pildančios 6 pav. Pavaizduotą IMEI ir MSISDN duomenų bazę, sudarymas. Prireikus sužinoti konkretaus MSISDN telefono modelį, sistema turėtų atlikti tokius veiksmus:

- Iš HLR pagal turimą MSISDN nustatyti IMSI;
- Iš EIR pagal turimą IMSI nustatyti IMEI;
- Negavus informacijos iš EIR, IMEI gavimui siųsti papildomą užklausimą MSC.

Norint užtikrinti greitą tokios sistemos atsakymo laiką bei siekiant minimaliai apkrauti operatoriaus sistemas, siūloma realizuoti duomenų bazės, saugančios MSISDN ir IMSI poras, kešavimą. Tokio būdu vieną kartą atlikus minėtus veiksmus ir sužinojus MSISDN atitinkantį IMEI numerį, pakartotinis HLR bei EIR apklausimas tam pačiam MSISDN galėtų vykti tik po tam tikro laiko intervalo.

### 5.1.2. IMEI ir telefono modelių duomenų bazė

Jau aptarėme, kad telefono modeliui nusakyti pakanka IMEI numerio pirmosios dalies, vadinamos TAC. Tiesa, TAC nusakys tik bendrą telefono modelio pavadinimą, be konkrečios

programinės įrangos versijos SV, tačiau ji dažniausiai ne žymiai įtakoja telefono modelio savybes.

Visų telefonų gamintojams iki šiol priskirtų TAC ir juos atitinkančių telefono gamintojų bei modelių duomenų bazė viešai nėra publikuojama, tačiau darbo metu buvo išsiaiškinta, kad GSM asocijacija [GSMA] savo nariams, tarp kurių yra ir daugelis Europos mobiliojo ryšio operatorių, suteikia prieigą prie pilnos TAC ir telefono modelių duomenų bazės. Ji yra vadinama akredituotų tipų duomenų baze (angl. Type Accreditation Database). Tai tekstinis failas, kuris susideda iš tokių stulpelių, vienas nuo kito atskirti vertikaliu brūkšniu „|“:

- TAC numerio
- FAC numerio
- Marketinginio telefono modelio pavadinimo
- Pažymėjimo tipo (angl. Designation type)
- Gamintojo
- Palaikomų dažnių
- Priskyrimo datos

Turint tokią duomenų bazę, belieka tik atlikti periodinius jos atnaujinimus. Tam, kad nereikėtų vėl iš naujo importuoti visos duomenų bazės, reikėtų imti tik tas TAC serijas, kurių priskyrimo data yra didesnė nei paskutinio atlikto šios bazės importavimo data.

## **5.2. Telefono modelio ir jį atitinkančio User-Agent duomenų bazė**

Norint nusakyti telefono modelio savybes, žinoti vien telefono modelio pavadinimą nepakanka. Kaip jau minėjome, telefonas WAP užklauso metu dažniausiai siunčia nuorodą į vartotojo naršyklės profilį UAprof specialioje HTTP antraštėje. Tačiau kaip žinoti nuorodą į UAprof žinant tik telefono modelio pavadinimą? Šiai problemai išspręsti reikalinga iš anksto turėti telefonų modelių ir jų savybių duomenų bazę.

Darbe kaip tik ir aptarėme universalią telefonų modelių ir jų savybių duomenų bazę WURFL. Šioje duomenų bazėje kiekvienas telefonas yra identifikuojamas vartotojo naršyklės agentu User-Agent. Siūlomoje naudoti GSM asociacijos IMEI TAC duomenų bazėje duomenų User-Agent nėra, tačiau yra įmanomas dalinis jo sukonstravimas iš gamintojo bei telefono modelio pavadinimų, kadangi paprastai User-Agent antraštėje visų pirma eina įrenginio gamintojas (pvz. Nokia), o vėliau pats modelis bei detalesni jo aprašymai (programinės įrangos versija ir pan.). Taigi, norint kuo tiksliau atlikti paiešką User-Agent

sąrašė turint telefono gamintoją bei modelį, reikalingas paieškos algoritmas. Pavyzdžiui, įrašas apie Nokia 6230 TAC GSM asociacijos duomenų bazėje atrodo taip:

*TAC/FAC/Marketing Name/Designation Type/Manufacturer/Bands Supported/Date*  
*352953/00/Nokia 6230/RH-12/Nokia Corporation/90018001900/15-Jan-2004*

O šį telefoną atitinkanti User-agent antraštė atrodo taip:

*Nokia6230/2.0 (04.44) Profile/MIDP-2.0 Configuration/CLDC-1.1*

Matome, kad šiuo konkrečiu atveju pakaktų iš „Marketing Name“ reikšmės išmesti tarpą ir atlikti paiešką WURLF duomenų bazėje. Tiesa, sukonstruota eilutė dar nebūtų pilnas User-Agent, todėl tektų ieškoti visame WURLF User-Agent medyje, ieškant pirmo tokio User-Agent, kurio pradžia „Nokia6230“. Tokios paieškos pagalba galime gauti tik „artimiausio“ telefono modelio savybes, kadangi paieškos metu nėra galimybės atsižvelgti į telefono programinės įrangos versija SVN, kuri nėra pateikiama GSMA TAC duomenų bazėje.

Kitas būdas sudaryti dalinį User-Agent – simbolių eilutė sukonstruoti iš „Manufacturer“ ir „Marketing Name“ junginio. Būtent tokiu principu yra konstruojami daugelis mobilių telefonų User-Agent, pvz. „SAMSUNG-SGH-X100/PEARL“, „SAGEM-myX-6/1.0“. Pirmos iteracijos metu nepavykus surasti nei vieno paieškos sąlygą tenkinančio User-Agent, procesą reikėtų tęsti kiekvienos iteracijos metu atimant po vieną raidę iš „Manufacturer“ galo. Tokiu būdu galima surasti daugelį „Siemens“ ar „Sony Ericsson“ modelių, kadangi būtent šių telefonų User-Agent konstruojami panašiu principu, pvz. Siemens CX65 atitinka „SIE-CX65/41“ ir pan.

Deja, aptartas paieškos algoritmas tinka tik kai kurių gamintojų telefonų modeliams. Taip yra todėl, kad GSMA TAC duomenų bazėje telefono modelio pavadinimą atitinkantis „Marketing Name“ ne visada yra vienareikšmis. Pavyzdžiui, Lietuvoje nemažą rinkos dalį užimančių „Motorola“ modelių C-350, C-450, C-550 mes duomenų bazėje nerastume, kadangi „Marketing Name“ daugelio „Motorola“ telefonų atveju atitinka ne marketinginį pavadinimą, o patvirtintą produkto numerį (angl. Product Approval Number). Darbo metu buvo nustatyta, kad „Motorola“ viešai nepublikuoja duomenų bazės, kurios pagalba žinant patvirtintą produkto numerį būtų galima nustatyti marketinginį telefono modelio pavadinimą, tačiau dalį šios informacijos galima rasti „Motorola“ tinklalapyje. Pavyzdžiui, C550 marketinginį pavadinimą atitinka MC3-41D21 patvirtintas produkto numeris:

*TAC/FAC/Marketing Name/Designation Type/Manufacturer/Bands Supported/Date*  
*352718/00/MC3-41D21/MC3-41D21/Motorola Inc./9001800/11-Nov-2003*

Su kitų žymesnių telefonų gamintojų įrašais TAC duomenų bazėje paininga yra dar didesnė. Pavyzdžiui, vieno ir to paties telefono modelio „Sony Ericsson T630“ vienu atveju gamintojas yra „Ericsson Mobile Comms AB“, o kitu – „Sony Ericsson Mobile Communications AB (Nya Vattentornet, Sweden)“, taigi aptartas paieškos algoritmas, kuris User-Agent sudaro iš gamintojo ir telefono modelio, čia taip pat ne visada tinka.

Atlikus GSMA TAC duomenų bazės bei mobiliųjų telefonų User-Agent formavimo analizę galima daryti išvadą, kad neegzistuoja vienareikšmis algoritmas, susiejantis telefono IMEI TAC su jo WAP naršyklės User-Agent antrašte.

Norint užtikrinti tokios duomenų bazės pilnumą, siūloma pasitelkti visus išvardintus šaltinius:

- GSMA TAC [GSMA-TAC] duomenų bazę;
- Nobbi TAC [TAC] duomenų bazę;
- WAP sesijos metu naršyklės siunčiamą User-Agent susieti su naudojamu IMEI.

Akivaizdu, kad aptariamoms duomenų bazės sudarymo kokybei užtikrinti reikalingas ir žmogiškasis faktorius – administratorius, kuris periodiškai patikrintų duomenų korektiškumą.

### **5.2.1. Telefono savybių pateikimas trečiosioms šalims**

Šiame poskyryje jau aptarėme, kaip būtų galima sudaryti telefonų modelių, kuriais naudojasi mobiliojo ryšio klientai bei jų savybių duomenų bazę, tačiau reikalingas būdas šiuos duomenis pateikti trečiosioms šalims.

Šiam tikslui siūlome naudoti vieną iš egzistuojančių operatoriaus sąsajų – PPG (angl., Push Proxy Gateway), kurio pagrindinė funkcija yra suteikti galimybę trečiosioms šalims nusiųsti vartotojui tam tikrą informaciją be jo atskiro pageidavimo.

PPG numato specialų užklauskos tipą, kuris yra skirtas vartotojo telefono įrenginio savybėms nustatyti (angl. Client Capabilities Query Service). Pagal PPG standartą, šios užklauskos palaikymas nėra privalomas, o kaip rezultatas – atsakymas į užklauską – turėtų būti pateikiamas vartotojo naršyklės profilis UAPProf arba jo fragmentas [Wap01].

Nors galimi ir kiti protokolai, kurių pagalba būtų galima perduoti informaciją apie telefono savybes, tačiau tokios informacijos pateikimas naudojant standartus užtikrintų didesnę šios paslaugos suderinamumą su kitomis sąsajomis.

### **5.3. Turinio adaptavimo platforma**

Praeituose šio darbo skyriuose aptarėme telefono modelio ir jo savybių atpažinimo galimybes bei galimus turinio adaptavimo modelius. Kartais pakanka telefono savybių atpažinimo ir turinio adaptavimo mechanizmus taikyti aplikacijos (pavyzdžiui, orų prognozių) lygyje, tačiau daugėjant telefonų, galinčių apdoroti gausybę skirtingų turinio tipų, skaičiui bei pačių paslaugų įvairovei, neišvengiamai reikalinga vieninga platforma, kuria galėtų pasinaudoti visos aplikacijos, kurioms reikalingi turinio mechanizmai.

Šiame poskyryje aptarsime OMA STI standarte apibrėžtą standartizuotos perkodavimo platformos interfeisą bei atvirojo kodo įrankius, kurių pagalba galima atlikti turinio transformacijas.

#### **5.3.1. Standartizuota perkodavimo platforma**

Galima sakyti, kad šiuo metu didžiausias poreikis standartizuotai perkodavimo platformai yra MMS centruose. MMS standartai numato pagrindinius turinio adaptavimo, kuriuos turi atlikti MMS centrus, mechanizmus, tačiau tai liečia tik bazinius MMS turinio tipus – tekstus, garsus ir paveikslėlius. Naujausi telefonai MMS kadruose gali atvaizduoti ir kur kas sudėtingesnius turinio tipus – vektorinę grafiką (SVG), video medžiagą, realaus laiko vaizdus (angl. streaming) ir pan. Dėl šios priežasties MMS centrui atsiranda būtinybė šias perkodavimo užduotis deleguoti specializuotai perkodavimo platformai [Bod05].

OMA pateikia standartizuotos perkodavimo platformos (angl. Standard Transcoding Interface) STI specifikaciją, kuri apibrėžia būdą, kaip daugialypės terpės aplikacijų platforma (angl. multimedia application platform) turėtų bendrauti su perkodavimo platforma tam, kad užtikrinti aukštą suderinamumą tarp skirtingų gamintojų platformų [Bod05].

Už OMA STI standartą yra atsakinga OMA BAC (angl. Browser and Content) – „naršyklės ir turinio“ darbo grupė. Šiai grupei yra pavesta specifiuoti aplikacijų, tokių kaip DRM bei PUSH, technologijas, standartizuotą perkodavimo interfeisą bei vartotojo naršyklės profilį.

OMA specifikacijos yra paremtos įgalinančiųjų (angl. enabler) ir suderinamųjų (angl. interoperability) leidinių pateikimu. Įgalinantysis leidinys – tai specifikacijų rinkinys, būtinas norint realizuoti paslaugas, tokias kaip MMS, skaitmeninių teisių valdymą (DRM), naršymą ir

pan. OMA apmatų (angl. draft) techninės specifikacijos yra prieinamos tik aljanso nariams ir jomis neturėtų remtis komerciniai sprendimai. Tik po to, kai techninės specifikacijos įgauna pakankamą brandą galimam taikymui komerciniuose sprendimuose, OMA viešai publikuoja tokių specifikacijų rinkinį – taip suformuojamas įgalinantysis leidinys [Bod05].

Nors atliekant darbą buvo remiamasi apmatine OMA STI (OMA-STI-V1\_0-20050209-D) standarto versija, šį interfeisą jau realizuoja keletas komercinių su turinio adaptavimu susijusių produktų – *Adamin* „*MEDIASPIRE 5.0*“ ir *Mobixell* „*Central RMSC*“. STI interfeisas taip pat yra įtraukiamas į naujesnes MMS architektūrą vaizduojančias diagramas.

Tolimesniuose skirsniuose aptarsime žymius ir nežymius turinio kokybės praradimus, turinio formatus bei STI interfeisą.

### **5.3.2. Žymūs ir nežymūs turinio praradimai**

Jeigu siuntėjo įrenginys siunčia vieną ar keletą turinio objektų (pvz. MMS žinutę), o dalies jų gavėjo įrenginys nepalaiko, atsiranda būtinybė tokius objektus transformuoti į gavėjo įrenginiui priimtina formą. Blogiausiu atveju, jei nustatoma, kad objektas neturi tinkamo atitikmens, t.y. yra netransformuojamas, tuomet toks objektas turėtų būti pašalintas (pvz. iš MMS žinutės). MMS žinučių atveju, turinio adaptavimas dažniausiai atliekamas atsiuntimo metu pagal vartotojo naršyklės savybes. OMA MMS suderinamumo dokumentas [OMA-ConfD] turinio adaptavimo užduotis apibrėžia priklausomai nuo praradimo lygių, taikomų daugybės terpės objektams.

#### **5.3.2.1. Nežymus praradimas (angl. minor degradation)**

Šiai praradimo rūšiai priklauso užduotis, kurios metu daugiausiai adaptuojamas žinutės dydis, paveikslėlio raiška, garso ir video kokybė atsižvelgiant į gavėjo įrenginį žymiai neiškraipant žinutės turinio. Nežymaus praradimo atveju, gavėjui pakeitimai žinutėje nėra ženkliai matomi.

#### **5.3.2.2. Žymus praradimas (angl. major degradation)**

Žymų praradimą sukelia tokia užduotis, kurios metu yra žymiai pakeičiamas žinutės turinys, pavyzdžiui, pašalinant visą turinio objektą iš žinutės ar konvertuojant video klipą į animuotą ar neanimuotą paveikslėlį.

### 5.3.2.3. Perkodavimo lentelės

[OMA-ConfDoc] dokumentas yra pateikia žymius ir nežymius turinio praradimus atspindinčios lentelės. 6-oje lentelėje pateiktas garsų perkodavimas, 7 – paveikslėlių, o 8 – video perkodavimas.

6 lentelė. Garsų perkodavimas.

Į: Iš:	AMR-NB	13K	SP-MIDI	Standartinis MIDI
AMR-NB	Nežymus	Nežymus	Negalimas	Negalimas
13K	Nežymus	Nežymus	Negalimas	Negalimas
SP-MIDI	Žymus	Žymus	Žymus	Negalimas
Standartinis pirmojo lygio MIDI	Žymus	Žymus	Žymus	Žymus

Į: Iš:	JPEG	GIF87a	GIF89a	WBMP
JPEG	Nežymus	Negalima	Negalima	Negalima
GIF87a	Nežymus	Nežymus	Negalima	Negalima
GIF89a	Žymus	Žymus	Nežymus	Negalima
WBMP	Žymus	Negalima	Negalima	Negalima

7 lentelė. Paveikslėlių perkodavimas.

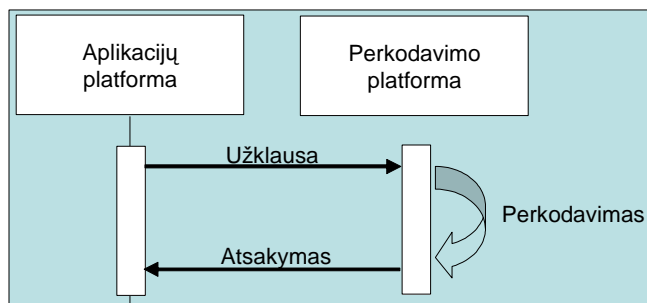
Į: Iš:	H263	MPEG4	JPEG	GIF87a	GIF89a
H263	Nežymus	Negalima	Žymus	Žymus	Žymus
MPEG4	Nežymus	Nežymus	Žymus	Žymus	Žymus

8 lentelė. Vaizdų perkodavimas.

Svarbu pastebėti, kad tik nuo MMS standarto 1.2 versijos MMS centrams (su išoriniu perkoduotoju ar be jo) yra privaloma palaikyti nežymią turinio adaptaciją. Žymios turinio adaptacijos palaikyti nereikalaujama.

### 5.3.3. Standartinis perkodavimo interfeisas

Bendravimas OMA STI interfeise tarp išorinių aplikacijų platformų ir perkoduotojo apibrėžiamas transakcinių užklausų ir atsakymų į jas principu taip, kaip parodyta 7 pav.



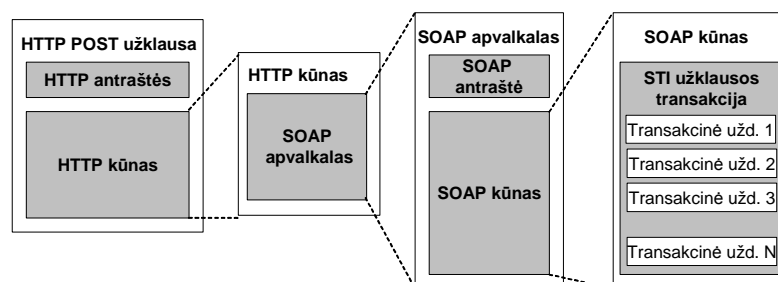
7 pav. STI perkodavimo operacija

Užklausa sudaro vieno ar kelių transakcinių užduočių (angl. transaction jobs) aprašai. Atsakymo metu kaip rezultatai gražinamos perkoduotojo įvykdytos transakcinės užduotys.

Užklausų ir atsakymų perdavimui STI numatytas SOAP protokolas, HTTP naudojant kaip transportavimo protokolą. SOAP formato pranešimai yra inkapsuliuojami į HTTP Post užklausas ir atsakymus.

#### 5.3.3.1. STI užklausos transakcija

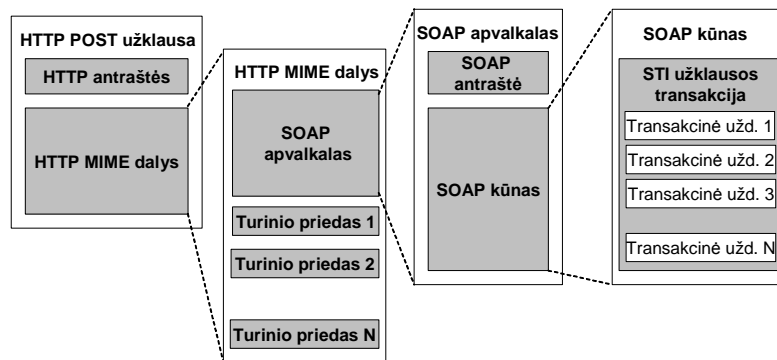
Užklausos transakcija išorinei aplikacijai leidžia išoriniam perkoduotojui užduoti vieną ar kelias transakcines užduotis. Užklausos transakcija kartu su perkodavimo instrukcijomis kiekvienai užduočiai yra inkapsuliuojama į SOAP apvalkalą (angl. envelope) kūną (angl. body). SOAP apvalkalas yra įtraukiamas į HTTP POST kūną taip, kaip parodyta 8 pav.



8 pav. STI užklausos transakcija be turinio priedų

Tokios transakcinės užklausos metu perkoduojami daugialypės terpės objektai nėra įtraukiami į užklausą. Transakcinių užduočių instrukcijose naudojamos nuorodos (URL) į

išorinę platformą, iš kurios objektus galės pasiekti perkoduotojas. Kita alternatyva yra į transakcinę užklausą įtraukti visus turinio objektus. Tokiu būdu visi turinio objektai kartu su SOAP apvalkalu yra įtraukiamos kaip MIME daugiadalės (angl. multipart) struktūros kūnai. Tokios užklausos pavyzdys pavaizduotas 9 pav.



9 pav. STI užklausos transakcija su turinio priedais

Transakcinės užklausos dalis – transakcinė užduotis – yra sudaryta iš dviejų sekcijų: sekcijos, aprašančios šaltinį (angl. source) (pvz. norimą transformuoti daugialypės terpės objektą, jo formatą ir pan.) bei kitos sekcijos, aprašančios tikslą (angl. target) – kokio rezultato tikimasi (priklausomai nuo vartotojo naršyklės profilio ir pan.).

Aplikacijų platforma (pvz., MMS centras) turinio adaptavimui gali naudoti du būdus. Pirmuoju būdu kiekvienas turinio objektas yra pateikiamas kaip atskira transakcinė užduotis. Antruoju būdu perkoduotojui kaip viena transakcinė užduotis yra pateikiama visa žinutė (pavyzdžiui, SMIL prezentacija) pridedant visos žinutėje esančius turinio priedus.

9-oje lentelėje yra pateikti svarbesni STI transakcinės užklausos parametrai, jų paaiškinimai, galimos tėvinės šakos bei ar pagal standartą jie yra privalomi.

Galimos tėvinės šakos	Pavadinimas	Privalomas / neprivalomas	Paaiškinimas
	<b>TranscodingRequest</b>	Privalomas (1..1)	Keletas transakcinių užduočių gali būti pateikta vienos užklausos metu.
TranscodingRequest, target	profileID	Neprivalomas (0..1)	Nustatytas profilis, kuris bus naudojamas visose šios užklausos transakcinėse užduotyse, nebent būtų perrašytas <i>transcodingParams</i> ar <i>profileID</i> transakcinės užduoties lygyje. Gali būti naudojama vartotojo antraštės (User-Agent) eilutė, vartotojo antraštės profilio (UAProf) eilutė (URL) arba privati (angl. proprietary) eilutė. Jei užklausoje pateiktas tikslinio įrenginio User-Agent arba UAProf, perkodavimo

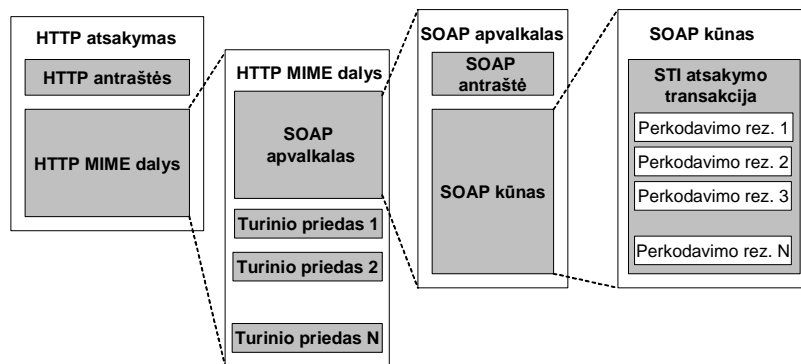
			platforma juos turėtų atpažinti. <i>profileID</i> reikšmė gali būti papildoma <i>applicationType</i> parametru.
TranscodingRequest, target	applicationType	Neprivalomas (0..1)	Aplikacija, kuriai atliekamas perkodavimas. Šis parametras gali būti naudojamas <i>profileID</i> papildymui, nurodant UAProf sekciją („Browsing“, „MMS“ ir pan.). Jei neperrašomas <i>target</i> lygyje, šis parametras galioja visiems užduočių transakcinėje užklausoje tikslams.
TranscodingRequest, transcodingJob	policyRef	Neprivalomas (0..1)	Tai URI į taisyklių sąrašą, kurios bus taikomos transakcinės užklausoje metu. Šio parametro standartas tiksliai neaprašo, tačiau taisyklės gali būti panaudojamos perkodavimo platformos elgsenai aprašyti (turinio elementų prioritetai, nustatytojo profilio panaudojimas ir pan.). Jei <i>policyRef</i> nurodytas tiek užklausoje, tiek užduoties lygiuose, <i>policyRef</i> užduoties lygyje turėtų turėti pirmenybę.
TranscodingRequest, transcodingJob	adaptationClasses	Neprivalomas (0..1)	Informacija apie adaptavimo klases – kokios turėtų būti naudojamos ir ar jos apskritai galimos. Jei <i>adaptationClasses</i> yra nurodytas transakcinės užduoties lygyje, jis turėtų turėti pirmenybę prieš tą, kuris yra nurodytas transakcinės užklausoje lygyje (jei nurodytas apskritai).
TranscodingRequest, transcodingJob	transcodingParams	Neprivalomas (0..1)	Perkodavimo parametrai, kurie, jei neperrašyti transakcinės užduoties lygyje, bus naudojami visoms užduotims. Kai transakcinėje užklausoje nurodytas profilis, <i>transcodingParams</i> reikšmės įgauna pirmenybę prieš atitinkamas reikšmes profilyje. Šiame elemente aprašomi perkodavimo parametrai ir apribojimai kiekvienam turinio tipui (tekstui, video, paveikslėliui, garsui ar daugiadaliui objektui): min. Ar maks. dydis baitais, turinio tipas, raiška, kanalų skaičius, išsidėstymas, vaizdo koduotės, dydis, plotis ir pan.
TranscodingRequest	transcodingJob	Privalomas (1..n)	Struktūra, aprašanti transakcinę užduotį. Šis elementas užklausoje gali pasikartoti keletą kartų.
TranscodingRequest, transcodingJob, target, source	extensionData	Neprivalomas (0..1)	Privačių plėtinių duomenys. Tai parametrai, kurie apibrėžti bendru sustarimu tarp aplikacijos ir perkodavimo platformų. Jei perkodavimo platforma neatpažįsta <i>extensionData</i> parametro reikšmės, jis turėtų būti ignoruojamas bei atsakymo metu apie tai suformuojamas išpėjimas. Šie parametrai taip pat gali turėti nuorodas į išorinę duomenų struktūrą ar priedus.
adaptationClasses	adaptationClass	Neprivalomas (0..n)	
adaptationClass	className	Privalomas	Adaptavimo klasės, kurios pageidauja

		(1..1)	aplikacija, pavadinimas.
adaptationClass	allowed	Neprivalomas (0..1)	Visai transakcinei užklausiai ar atskirai užduočiai nurodo, kokios adaptacijos, priklausančios nurodytai adaptacijos klasei, yra leidžiamos. Jei <i>allowed</i> parametras nenurodomas, pagal nutylėjimą adaptacijos klasė yra leidžiama. Jei nurodoma konkrečiai užduočiai, prieštaravimo tarp leistinų ir neleistinų adaptacijų atveju pirmenybė yra teikiama neleistinoms adaptacijoms. Kitaip tariant, adaptavimo operacija yra draudžiama, jei ji atitinkamą transakcinę užduotį priskirtų adaptavimo klasei, kuri yra draudžiama (pvz. veiksmas, leidžiamas nežymioje, bet draudžiamas žymioje klasėje bus draudžiamas).
adaptationClass	classRef	Neprivalomas (0..1)	URI į nustatytosios adaptavimo klasės aprašą.
transcodingJob	<b>source</b>	Privalomas (1..1)	Transakcinės užduoties šaltinis
transcodingJob	<b>target</b>	Neprivalomas (0..1)	Šios transakcinės užduoties tikslas.
source	contentType	Privalomas (1..1)	Šaltinio turinio tipas
source	<b>contentTypeParams</b>	Neprivalomas (0..1)	Šaltinio turinio tipo parametrai (pvz. koduotė)
source	location	Privalomas (1..1)	Tiems šaltiniams, kurie neįtraukti į transakcinę užklausą, nurodomas pilnas kelias (URL) į išorinę saugyklą kartu su resurso pavadinimu. Šaltiniams, kurie įtraukti į užklausą, nurodomas priedo užklausoje turinio ID (cid) [RFC2392].
target	<b>externalLocation</b>	Neprivalomas (0..1)	Nuoroda į išorinį šaltinį. Perkodavimo platforma <i>jobResult output</i> elemente turėtų gražinti pilną perkoduoto resurso URI. Jei šis parametras nenurodytas, perkoduotas turinys bus gražinamas kaip priedas, įtrauktas į transakcinį atsakymą (net jei buvo naudojama nuoroda į šaltinio elementą).

9 lentelė. Svarbesni STI transakcinės užklausoje parametrai.

### 5.3.3.2. STI atsakymo transakcija

Po to, kai aplikacijų platforma pateikė transakcinę užklausą, perkodavimo platforma įvykdo transakcinę užduotį ir pateikia perkodavimo rezultatus atgal į aplikacijų platformą kaip *atsakymo transakciją*. Ši atsakymo transakcija yra inkapsuluota HTTP atsakyme į atitinkamą HTTP POST užklausa pateiktą užklausoje transakciją. Atsakymo transakcijos struktūra pateikta 10 pav.



10 pav. STI atsakymo transakcija

Didėle dalis atsakymo transakcijos parametrų yra panašūs į užklauso transakcijos parametrus. Svarbiausieji jų pateikti 10-oje lentelėje.

Galimos tėvinės šakos	Pavadinimas	Privalomas / Neprivalomas	Paaiškinimas
	<b>TranscodingResponse</b>	Sąlyginis (0..1)	Atsakymo duomenys. Šis elementas privalo egzistuoti, jei operacija buvo sėkminga ir turėtų egzistuoti, jei operacijose buvo klaidų.
TranscodingResponse, jobResult	<b>mainReturnResult</b>	Privalomas (1..1)	Operacijos gražinamas rezultatas
TranscodingResponse, jobResult	<b>additionalReturnResults</b>	Neprivalomas (0..1)	Be <i>mainReturnResult</i> , atsakymo transakcija ar užduotis gali gražinti papildomų rezultatų.
TranscodingResponse	totalDuration	Privalomas (1..1)	Laikas milisekundėmis, kurį perkodavimo platforma sugaišo visos užduoties atlikimui. Šis skaičius yra visos operacijos nuo tada, kai ji pasiekė platformą iki tada, kai buvo gražintas atsakymas, laikas. Jis gali būti skirtingas nei visų užduočių trukmių suma (arba didesnė reikšmė, rodanti visos operacijos kaštus, arba mažesnė reikšmė, jei kelios užduotys buvo atliktos lygiagrečiai).
TranscodingResponse	<b>jobResult</b>	Sąlyginis (0.. n)	Užduočių rezultatai. Gali būti daugiau nei vienas (vienas rezultatas vienai transakcinei užduočiai). Šis elementas privalo egzistuoti, jei <i>TranscodingResponse mainReturnResult</i> yra ne klaida.
jobResult	<b>adaptationsPerformed</b>	Neprivalomas (0..1)	Adaptavimo klasės, susietos su realiai atliktu turinio adaptavimu.
jobResult	duration	Neprivalomas (0..1)	Laikas, kurį perkodavimo platforma užtrukto atliekant užduotį.
jobResult	<b>output</b>	Sąlyginis	Informacija apie gautą rezultatą.

		(0..1)	Šis elementas privalo egzistuoti, jei užduoties <i>mainReturnResult</i> yra ne klaida.
<b>adaptationsPerformed</b>	adaptationPerformed	Neprivalomas (0..n)	Adaptavimo klasė, susieta su realiai atliktu turinio adaptavimu. Perkodavimo platforma turėtų gražinti sąrašą visų adaptavimo klasių, esančių transakcinėje užklausoje ir sutiktų perkodavimo proceso metu kiekviename <i>adaptationPerformed</i> elemente.
output	contentType	Privalomas (1..1)	Gražinamo rezultato turinio tipas.
output	mediaSize	Neprivalomas (0..1)	Realus objekto dydis.

9 lentelė. Svarbesni STI transakcinio atsakymo parametrai.

### 5.3.3.3. STI privalumai ir trūkumai

Pats OMA aliansas STI apibrėžia šiuos pagrindinius STI, kaip standartizuoto standarto sukūrimo, privalumus:

- Naujos architektūros atsiradimo galimybė ten, kur perkodavimo sistema naudojami įvairūs servizai operatoriaus pusėje;
- Suteikia galimybę operatoriams ir paslaugų tiekėjams pasirinkti geriausią daugialypės terpės adaptavimo sistemą tokiems terminalams ir formatams, kurių palaikymas dar tik planuojamas;
- Apibrėžia terminalinės vartotojo naršyklės informaciją, reikalingą kokybiškai adaptacijai iš vieno įrenginio į kitą;
- Standartizuotas interfeisas apibrėš su aplikacija ar paslauga susietas perkodavimo taisyklės ar patarimus (angl. hints) tam tikram kanalui (įrenginio tipui, modalumui ir pan.).

OMA STI interfeisas, be to, kad tai yra vienas pirmųjų standartų, apibrėžiančių turinio adaptavimo procesą, turi ir kitų privalumų. SOAP protokolas pateikia lankstų mechanizmą žinučių (pavyzdžiui, užklauso transakcijos) praplėtimą decentralizuotu ir modulinio būdu be papildomų žinių apie bendraujančias šalis. Tipiniai tokių plėtinių pavyzdžiai, realizuojami kaip SOAP antraštės, gali būti autentifikacija, transakcijų valdymas, apmokestinimo informacija ir pan. SOAP taip pat apibrėžia keletą atributų, kurie gali būti panaudoti

indikacijai, kas turėtų apdoroti tam tikrą savybę ir ar toks apdorojimas yra privalomas. [OMA-STI]

Kita labai svarbi OMA STI savybė – lankstumas. Adaptuojant turinį nėra prisirišama vien tik prie vartotojo antraštės profilio UAProf – pageidaujamas savybes bei apribojimus turinio adaptavimui galima nurodyti adaptavimo klasių ir perkodavimo parametrų pagalba. Tai yra ypač aktualu tokioms paslaugoms, kurių turiniui turi būti išpildyti tam tikri reikalavimai. Pavyzdžiui, MMS orų prognozė kas diena siunčia video reportažą apie rytdienos orus. Šiuo atveju aplikacijų platforma, atliekanti MMS žinutės išsiuntimą prenumeratoriams, perkodavimo platformai galėtų prioriteto tvarka išvardinti, kad geriausia alternatyva video failams yra animuotas GIF arba keli įprasti GIF ar JPEG paveikslėliai, gauti skaidant video failą kas vieną sekundę.

Kaip vieną iš OMA STI interfeiso trūkumų būtų galima paminėti tai, kad didžioji dalis elementų yra neprivalomi. Tai palieka tam tikrą neapibrėžtumą galimose šio interfeiso realizacijose. Šis interfeisas skirtingose realizacijose gali būti panaudotas skirtingai taip pat dėl „laisvų“ elementų reikšmių bei dėl minėto *extensionData* elemento, kuriame gali būti perduodama standarte nenumatyta informacija.

Dar vienas labai svarbus aspektas, su kuriuo tektų susidurti įgyvendinant OMA STI standarto realizaciją – turinio transformavimo įrankiai. Interfeisas numato tiek vieno turinio formato transformavimą į kitą, tiek turinio objekto savybių keitimą, tačiau nėra atsižvelgiama, ar tikrai visada tokios transformacijos yra įmanomos. Dažniausios tai sąlygojančios priežastys gali būti formatų nesuderinamumas bei tam tikri licenziniai apribojimai, dėl kurių negalima atlikti vienokių ar kitokių transformacijų. Kitame šio skyriaus poskyryje kaip tik ir aptarsime keletą turinio transformavimo įrankių bei apribojimus juose.

Nepaisant kelių paminėtų trūkumų, OMA STI standartas yra didelis žingsnis į priekį siekiant sukurti standartizuotą interfeisą turinio adaptavimui. Dėl gausėjančio paslaugų ir naujų turinio tipų, tokios platformos, realizuojančios STI, pritaikymas jau šiandieną yra beveik neabejotinas.

#### **5.4. Atvirojo kodo įrankiai turinio transformacijai**

Šiame poskyryje aptarsime atvirojo kodo įrankius, kurių pagalba galima atlikti populiariausių turinio tipų, naudojamų mobiliuosiuose telefonuose, transformacijas bei savybių pakeitimą.

### 5.4.1. Paveikslėlių transformavimas

Vienas iš populiariausių atvirojo kodo įrankių paveikslėlių manipuliacijai ir konvertavimui yra *ImageMagick* [IMAGICK]. Šis įrankis palaiko daugiau nei 90 įvairių formatų, tarp kurių yra WAP standartuose numatytas nespalvoto paveikslėlio formatas WBMP, bei gausiai mobiliuosiuose telefonuose naudojami GIF, JPEG, PNG formatai ir daugelis kitų. *ImageMagick* taip pat palaiko jau kai kuriuose telefonuose naudojamą vektorinės grafikos formatą SVG, tačiau dėl pačio formato sudėtingumo, jo palaikymas *ImageMagick* įrankyje dar nėra pilnas.

*ImageMagick* gali pasitarnauti ne tik norint konvertuoti vieno formato paveikslėlį į kitą – šio įrankio pagalba galima keisti įvairias paveikslėlio savybes: aukštį, plotį, raišką, kontrastą, spalvų kiekį ir pan. *ImageMagick* pagalba taip pat galima animuotą GIF paveikslėlį suskaidyti į keletą įprastų (neanimuotų) GIF, JPEG ar kito formato paveikslėlių. Tai labai aktualu, jei atliekam transformacijas iš video failo.

### 5.4.2. Garsų transformavimas

Mobiliuosiuose telefonuose grojamus garsus ir jiems naudojamus formatus galima suskirstyti į šias kategorijas:

- Sintetiniai garsai (monofoninės ir polifoninės MIDI melodijos)
- Kalba (plataus kanalo AMR-WB ir siauro kanalo AMR-NB, WAV)
- Muzika (AAC, MP3, AMR-NB, AMR-WB, WAV)

Dėl formato kompatišškumo, labiausiai paplitę ir daugiausiai palaikomi yra MIDI bei SP-MIDI formatai monofoninėms ir polifoninėms melodijoms bei AMR-NB kalbai bei muzikai.

AMR failams užkoduoti ir dekoduoti yra reikalingi 3GPP ANSI C kalba pateikiami ir laisvai platinami kodavimo algoritmai [3GPP-26.104].

WAV, MP3 ir AAC failų operacijoms atlikti paprastai naudojamas šių atvirojo kodo įrankių rinkinys: *mpg123* [MPG123], kurio pagalba galima konvertuoti MP3 failus į WAV ir atvirkščiai bei garsų manipulatorius *sox* [SOX], kuris palaiko AAC ir daugelį kitų garsinių formatų.

### 5.4.3. Video transformavimas

Mobiliuosiuose telefonuose labiausiai paplitęs video formatas – 3GPP failo formatas (3GP), paremtas MPEG-4 ir H.263 standartais. Šiais ir kitais video failų formatais sėkmingai

operuoja *ffmpeg* [FFMPEG] atvirojo kodo įrankis, kurio pagalba taip pat galima video failus konvertuoti į paveikslėlių rinkinį bei atlikti aptarto garsinio formato AMR transformacijas.

Dėl palaikomų formatų gausos ir lankstumo, *ffmpeg* naudoja daugelis kitų atvirojo kodo įrankių, pavyzdžiui, daugelybės terpės grotuvai *MPlayer*, *xine*, *VLC*, *avifile*, *gststreamer* ir kiti.

#### 5.4.4. Licenziniai apribojimai

Savo programose naudojant atvirojo kodo įrankius, labai svarbu atsižvelgti į tai, kokios licenzijos laikosi naudojamas įrankis. Mūsų aptarti, o taip pat ir daugelis kitų atvirojo kodo įrankių naudoja GPL ar su ja suderinamą licenziją, o tai reiškia, kad kompiliuoti savo programas su tokiomis bibliotekomis galėsime tik tuo atveju, jei mūsų programos licenzija bus taip pat suderinama su GPL. Vienas iš būtų šiai problemai spręsti yra įrankio panaudojimas ne kviečiant bibliotekos funkcijas, o naudojant „pipe“ tipo filtrus, t.y. perduoti įrankiui įeities failą, o iš jo gauti išeities failą. Tokiu būdu kuriamai programai iškeliamas reikalavimas, kad naudojami įrankiai turėtų būti iš anksto įdiegti į operacinę sistemą.

Kitas svarbus dalykas – tai patentai, ribojantys tam tikrų turinio formatų užkodavimo ar atkodavimo galimybes. Vienas populiariausių pavyzdžių – GIF formato suspaudimas. MPEG licenzijų įstaiga (LA – angl. Licensing Authority) taip pat stengiasi riboti bet kokių komercinių produktų, naudojančių MPEG užpatentuosius algoritmus, bet neturinčių atitinkamų licenzijų, išleidimą į rinką.

Pastaruoju metu taip pat labai daug sumaišties sukėlė Europos Komisijos planuojamas programinės įrangos patentų patvirtinimas, kas iš esmės labai suvaržytų daugelio šiuo metu veikiančių programų tolimesnį naudojimą bei vystymą. Daugelis daugialypės terpės (angl. multimedia) algoritmų turi vienokius ar kitokius patentus ir dėl šios priežasties aptartų atvirojo kodo įrankių panaudojimas turinio transformacijoms atlikti gali tapti komplikuoatas.

#### 5.5. WML/XHTML puslapiavimas

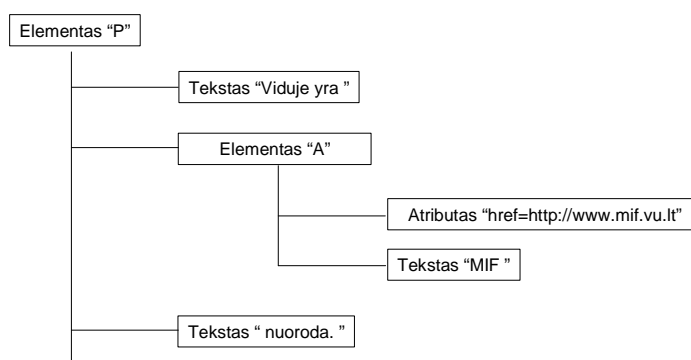
Dar viena nepaliesta problema – tai WML ar XHTML puslapiavimas. Šiai problemai spręsti netinka aptarti turinio adaptavimo mechanizmai dėl paprastos priežasties – informacijos puslapiavimas, t.y. skaidymas jos į kelis mažesnius segmentus, labai priklauso nuo konteksto.

Per telefono WAP naršyklę naršomus didesnius puslapius būtina puslapiuoti dėl WAP deko dydžio apribojimų. Kaip žinia, WML puslapis susideda iš deko, o pastarasis – iš vienos ar kelių WML kortelių. Vartotojo naršyklės profilyje UAProf maksimalus deko dydis yra nurodomas *WmlDeckSize* parametru *WapCharacteristics* sekcijoje. Šis parametras nurodo,

koks yra maksimalus (WBXML formatu, jei telefonas nepalaiko WAP 2.0) deko, kurį telefono naršyklė gali atvaizduoti, dydis.

Atlikus tipinių WML puslapių analizę paaiškėjo, kad dinaminis turinys dažniausiai būna tik puslapio centrinėje dalyje – antraštė (angl. header) ir baigiamoji dalis (angl. footer) paprastai nekinta. Todėl nuspręsta puslapiavimo mechanizmą taikyti tik dinaminiai puslapio daliai.

Atliekant realizaciją, nuspręsta pasinaudoti DOM modeliu, kurio pagalba bet kokį XML poaibio dokumentą galima susivesti į tam tikrą hierarchinį medį. Pavyzdžiui, HTML ištrauka „`<P>Viduje yra <A href=`”<http://www.mif.vu.lt>”`</a> MIF</a> nuoroda.</P>`“ transformuojasi į 11 pav. pavaizduotą DOM medį.



11 pav. DOM medžio pavyzdys.

Svarbu pastebėti, kad, pavyzdžiui, „A“ elementas yra neskaidomas, todėl dar vadinamas atominiu.

Tam, kad supaprastinti puslapiavimo logiką, buvo įsivesti du papildomi elementai, kurie turėtų gauti dinaminį turinį – tai sekcija arba elementas. Sekcija naudojama tais atvejais, kai turim sekcijos antraštę ir kūną, kurie vienas nuo kito negali būti atskirti, o tai reiškia, kad turi būti parodyti būtinai viename puslapyje.

Dinaminės puslapio dalies turinys turi tenkinti tokius apribojimus:

- Duomenys turi būti validaus XML formatu (XML antraštė nebūtina);
- Duomenys gali būti priskiriami elementui `<element>` arba `<section>`
- `<section>` elementas privalo turėti ir `<sectionHeader>`, ir `<sectionBody>` elementus
- Jei elementas `<element>` yra skaidomas, tai turi būti nurodoma atributui `type` priskiriant reikšmę `breakable`.

Realizuotas puslapiavimo mechanizmas veikia pagal tokią logiką:

- Jei <sectionHeader> arba <element> elementas netilptų į tuščią puslapį, jis prikabinamas einamajame puslapyje (fallback principas).
- Jei <element> yra pirmas sekcijos elementas, bet nebetelpa į einamąjį puslapį, jis priverstinai patalpinamas.
- Jei <element> elemento tipas yra *breakable* ir visas elementas nebetelpa į puslapį, bandoma patalpinti po vieną jame esančią šaką (XHTML elementą) Elemente esantys kiti HTML elementai nėra skaidomi. Jei netelpa šaka ir ji savo kūne turi tekstą, jis yra skaidomas į žodžius (atskirtus vienu tarpu), kiek jų telpa į einamąjį puslapį.

Puslapiavimo mechanizmas buvo realizuotas PHP programavimo kalba naudojant DOMXML biblioteką. Dėl nemažos viso puslapiavimo mechanizmo realizacijos apimties, antrajame priede pateiktos tik dvi f-jos: elemento į puslapį pridėjimui bei XML elemento (su atributais) aprašo dydžio paskaičiavimas.

## 5.6. Galimos alternatyvos

Šiame darbe nagrinėti turinio adaptavimo mechanizmai tinka tokioms paslaugoms, kurios bendrauja su mobiliuoju įrenginiu SMS ar MMS žinutėmis arba WAP naršyklės pagalba. Kiekvienas telefono įrenginys turi identifikacinį modulį SIM (angl., Subscriber Identity Module), kuris yra ne kas kitas, o lustas, dar vadinamas gudriąja kortele (angl. smart card). Europos telekomunikacijų standartizavimo institutas ETSI yra pateikęs standartus, aprašančius programinę sąsają (API), kurios pagalba galima kurti programas, veikiančias ne telefone, o SIM kortelėje. Pastaruoju metu ypač populiarėjo SIM kortelių, kuriose įdiegta Javacard technologija, o tai reiškia, kad programos SIM kortelei gali būti rašomos Java kalba.

Paslaugų pateikimo per SIM kortelę kaip tarpininką didžiausias privalumas – tarp telefono ir SIM kortelės esanti sąsaja yra pakankamai griežtai standartizuota, todėl tokiu būdu galima užtikrinti ypač didelį paslaugų suderinamumo lygį.

SIM kortelių technologijų nagrinėjimas netelpa į šio darbo rėmus, todėl šios temos nenagrinėsime.

Šiame skyriuje pateikėme projektinius sprendimus, kaip būtų galima realizuoti telefono modelio ir savybių atpažinimo sistemą bei turinio perkodavimo platformą. Ypač daug dėmesio skyrėme pirmajam standartui, kuris apibrėžia standartizuotą perkodavimo interfeisą. Skyriuje taip aptarta WML puslapiavimo problematika ir pateikti galimi jos sprendimo būdai.

## Išvados

Darbo metu buvo išnagrinėti pagrindiniai telefono modelio atpažinimo būdai, aptartos telefono savybių nustatymo galimybės bei pateikti kelių turinio adaptavimo mechanizmų esminiai principai. Atlikta analizė leido nusakyti kiekvieno iš būdų ar mechanizmų privalumus bei trūkumus. Teorinė analizė ir požiūris, nedetalizuojant atskirų turinio formatų, jų kodavimo ar transformavimo algoritmų, leido geriau įsigilinti į aptariamą problemą bei jos sprendimo būdus.

Darbe pateikiami siūlomi architektūriniai sprendimai mobiliojo įrenginio atpažinimui ir jo savybių nustatymui bei turinio adaptavimo platformai, kurie remiasi IMEI TAC kodų ir WURFL telefonų savybių duomenų bazėmis bei OMA siūlomu standartizuotu perkodavimo interfeisu STI. Pastarasis standartas yra bene pirmasis žingsnis į priekį sprendžiant mobilaus turinio problemą standartizuotu keliu, todėl šiam standartui išanalizuoti buvo skirta nemažai dėmesio.

Galima pakankamai drąsiai daryti išvadą, jog būtent standartai – jų įvairovė ir nesilaikymas – yra vienas esminių faktorių, lemiančių turinio adaptavimo problematiką, kuri, bet kita ko, ne tiek daug skiriasi nuo gerai žinomų problemų, kylančių dėl interneto naršyklių skirtumų. Dėl šios priežasties buvo būtina atlikti detalią nagrinėjamą probleminę sritį – turinio adaptavimo skirtingiems mobiliams įrenginiams – analizę.

Svarbu paminėti, jog darbo metu pasiūlyti sprendimai mobiliojo įrenginio atpažinimui ir jo savybių nustatymui bei turinio adaptavimo platformai remiasi jau egzistuojančiais standartais bei priemonėmis, kurių panaudojimas šios problemos sprendimui ne tik užtikrina didesnę suderinamumą su kitomis sistemomis, bet ir sudaro geresnes sąlygas tokio sprendimo tolimesnei plėtrai.

Manome, jog praktikoje taikant aptartus mechanizmus, galima pasiekti įtikinamų rezultatų didinant vartotojų, naudojančių mobiliąsias paslaugas, pasitenkinimo lygį. Tai ir buvo šio darbo tikslas.

## **Summary**

### **Content adaptation for different mobile devices**

Content adaptation is becoming the major issue in today's mobile solutions. The rapid growth of different content types in current and upcoming mobile services is a challenge both for mobile telecoms and content or service providers. Therefore ensuring high user experience is a difficult and often very expensive task.

This thesis discusses various methods for mobile device and its capabilities recognition. Different strategies of content adaptation are also deeply covered. The big part of this thesis is dedicated for possible architectural as well as technological ideas and solutions to design the systems of device recognition, its capabilities negotiation and the transcoding platform. GSM IMEI TAC and WURFL capabilities database and OMA STI standard were chosen as a base for the given solutions.

## Literatūros ir šaltinių sąrašas

- [Bod05] G. Le Bodic, Mobile Messaging, SMS, EMS and MMS, 2nd edition. Wiley&Sons, February 2005.
- [LL02] Wai Yip Lum; Lau, F.C.M. A context-aware decision engine for content adaptation. IEEE Pervasive Computing, Volume 1, Issue 3, pp. 41-49, 2002.
- [Son04] TeliaSonera. Web Content Adaptation. White Paper.  
URL: <http://www.medialab.sonera.fi/workspace/WebContentAdaptationWP.pdf>. August 2004.
- [Wap99] WAP User Agent Profile, WAP Forum, November 1999.
- [Wap01] WAP push architectural overview, WAP Forum, May 2001.
- [OMA-STI] OMA Standard Transcoding Interface specification.
- [OMA-ConfD] OMA Multimedia Messaging Service Conformance Document Candidate Version 1.2.
- [NOBBI-TAC] Nobbi. Database of Manufacturer, Model and 'Type Approval Code'.  
URL: <http://www.nobbi.com/tacquerydb.htm>.
- [GSMA-TAC] GSM asociacijos kredituotų tipų (IMEI TAC) duomenų bazė.  
URL: <https://infocentre.gsm.org/cgi-bin/tad.cgi>. Prieinama tik asociacijos nariams.
- [WURLF] Wireless Universal Resource File. URL: <http://wurfl.sourceforge.net/>.
- [RDF] W3C. Resource Description Framework. URL: <http://www.w3.org/RDF/>.
- [TSGS#19] Technical Specification Group Services and System Aspects. Meeting #19, Birmingham, UK, 17-20 March 2003.  
URL: [http://www.3gpp.org/ftp/tsg\\_sa/TSG\\_SA/TSGS\\_19/Docs/PDF/SP-030133.pdf](http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_19/Docs/PDF/SP-030133.pdf)
- [3GPP-26.104] 3GPP TS 26.104, ANSI-C code for the floating-point Adaptive Multi-Rate (AMR) speech codec.
- [IMAGICK] Paveikslėlių manipulatorius. URL: <http://www.imagemagick.org/>.
- [FFMPEG] Video manipulatorius. URL: <http://ffmpeg.sourceforge.net/>.
- [SOX] Garsų manipulatorius. URL: <http://sox.sourceforge.net/>.
- [MPG123] Garsų manipulatorius. URL: <http://www.mpg123.de/>.

## 1 priedas. Nokia 6230 vartotojo naršyklės profilio UAProf fragmentas.

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:prf="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschem-20021212#"
  xmlns:mms="http://www.wapforum.org/profiles/MMS/ccppschem-20010111#">
  <rdf:Description rdf:ID="Nokia6230">
    <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschem-
          20021212#HardwarePlatform" />
        <prf:BluetoothProfile>
          <rdf:Bag>
            <rdf:li>Generic Access Profile</rdf:li>
            <rdf:li>Service Discovery Application Profile</rdf:li>
            <rdf:li>Serial Port Profile</rdf:li>
            <rdf:li>Headset profile</rdf:li>
            <rdf:li>Dial up networking</rdf:li>
            <rdf:li>Object push Profile</rdf:li>
            <rdf:li>File transfer profile</rdf:li>
            <rdf:li>handsfree profile</rdf:li>
            <rdf:li>CAR SIM Access Profile</rdf:li>
            <rdf:li>lanAccess</rdf:li>
          </rdf:Bag>
          </prf:BluetoothProfile>
          <prf:BitsPerPixel>16</prf:BitsPerPixel>
          <prf:ColorCapable>Yes</prf:ColorCapable>
          <prf:ImageCapable>Yes</prf:ImageCapable>
          <prf:InputCharSet>
            <rdf:Bag>
              <rdf:li>ISO-8859-1</rdf:li>
              <rdf:li>US-ASCII</rdf:li>
              <rdf:li>UTF-8</rdf:li>
              <rdf:li>ISO-10646-UCS-2</rdf:li>
            </rdf:Bag>
            </prf:InputCharSet>
            <prf:Keyboard>PhoneKeypad</prf:Keyboard>
            <prf:Model>6230</prf:Model>
            <prf:NumberOfSoftKeys>3</prf:NumberOfSoftKeys>
            <prf:OutputCharSet>
              <rdf:Bag>
                <rdf:li>ISO-8859-1</rdf:li>
                <rdf:li>US-ASCII</rdf:li>
                <rdf:li>UTF-8</rdf:li>
                <rdf:li>ISO-10646-UCS-2</rdf:li>
              </rdf:Bag>
              </prf:OutputCharSet>
            <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
            <prf:ScreenSize>128x128</prf:ScreenSize>
            <prf:ScreenSizeChar>18x5</prf:ScreenSizeChar>
            <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
            <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
            <prf:TextInputCapable>Yes</prf:TextInputCapable>
            <prf:Vendor>Nokia</prf:Vendor>
            <prf:VoiceInputCapable>Yes</prf:VoiceInputCapable>
          </rdf:Description>
        </prf:component>
      </prf:component>
      <rdf:Description rdf:ID="SoftwarePlatform">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschem-
          20021212#SoftwarePlatform" />
        <prf:AcceptDownloadableSoftware>Yes</prf:AcceptDownloadableSoftware>

```

```

<prf:AudioInputEncoder>
<rdf:Bag>
<rdf:li>AMR</rdf:li>
</rdf:Bag>
</prf:AudioInputEncoder>
<prf:CcppAccept>
<rdf:Bag>
<rdf:li>application/vnd.wap.connectivity-wbxml</rdf:li>
<rdf:li>application/vnd.wap.sic</rdf:li>
<rdf:li>application/vnd.wap.slc</rdf:li>
<rdf:li>text/x-vCard</rdf:li>
<rdf:li>text/x-vCalendar</rdf:li>
<rdf:li>application/vnd.wap.hashcd-certificate</rdf:li>
<rdf:li>application/vnd.wap.signed-certificate</rdf:li>
<rdf:li>application/vnd.wap.cert-response</rdf:li>
<rdf:li>application/x-wap-prov.browser-bookmarks</rdf:li>
<rdf:li>text/html</rdf:li>
<rdf:li>text/vnd.wap.wml</rdf:li>
<rdf:li>application/xhtml+xml</rdf:li>
<rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
<rdf:li>application/vnd.oma.dd+xml</rdf:li>
<rdf:li>application/vnd.oma.drm.message</rdf:li>
<rdf:li>application/x-wallet-appl.user-data-provision</rdf:li>
<rdf:li>image/gif</rdf:li>
<rdf:li>image/jpeg</rdf:li>
<rdf:li>image/jpg</rdf:li>
<rdf:li>image/bmp</rdf:li>
<rdf:li>image/png</rdf:li>
<rdf:li>image/vnd.wap.wbmp</rdf:li>
<rdf:li>image/vnd.nok-wallpaper</rdf:li>
<rdf:li>video/3gpp</rdf:li>
<rdf:li>audio/3gpp</rdf:li>
<rdf:li>video/mp4</rdf:li>
<rdf:li>audio/amr</rdf:li>
<rdf:li>application/sdp</rdf:li>
<rdf:li>audio/midi</rdf:li>
<rdf:li>audio/mid</rdf:li>
<rdf:li>audio/x-midi</rdf:li>
<rdf:li>audio/x-mid</rdf:li>
<rdf:li>audio/sp-midi</rdf:li>
<rdf:li>application/vnd.nokia.ringing-tone</rdf:li>
<rdf:li>audio/mpeg</rdf:li>
<rdf:li>audio/mp4</rdf:li>
<rdf:li>audio/mpeg4</rdf:li>
<rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>
<rdf:li>application/vnd.oma.drm.content</rdf:li>
<rdf:li>application/vnd.met.ticket</rdf:li>
<rdf:li>application/java</rdf:li>
<rdf:li>application/vnd.oma.drm.rights+xml</rdf:li>
<rdf:li>application/vnd.met.receipt</rdf:li>
<rdf:li>application/java-archive</rdf:li>
<rdf:li>application/vnd.oma.drm.rights+wbxml</rdf:li>
<rdf:li>image/vnd.nok-oplogo-color</rdf:li>
<rdf:li>application/x-java-archive</rdf:li>
</rdf:Bag>
</prf:CcppAccept>

```

## 2 priedas. WML puslapavimui naudojamoms *addElement* ir *getNodeSize* f-jos

```

function addElement($node, $reserved = 0) {

    $buf = null;
    $size = 0;

    $elType = $node->get_attribute("type");

    if ($this->fitsToCurrentPage($node, $reserved) || ($elType != "breakable" && !$this->fitsToEmptyPage($node,
    $reserved))) {
        if (!$this->fitsToEmptyPage($node, $reserved)) {
            $this->fallback = true;
            $this->nextPage = true;
        }
        return $this->appendNode($node, false, true);
    }

    if ($elType == "breakable") {
        foreach ($node->child_nodes() as $child) {
            $nodeType = $child->node_type();
            if (
                $this->fitsToCurrentPage($child, $reserved + $this->len($buf)) ||
                ($nodeType != XML_TEXT_NODE && !$this->fitsToEmptyPage($child, $reserved))
            ) {
                $buf .= $this->appendNode($child, true, true);
                $child->unlink_node();
                // if (!$this->fitsToEmptyPage($child, $reserved)) $this->fallback = true;
            } else if ($child->node_type() == XML_TEXT_NODE) {
                $maxsize = $this->getMaxSize() - $this->len($buf) - $reserved - $this->getSize();
                $data = $this->dom->dump_node($child);
                if ($maxsize > 0 && $words = $this->getWords($data, $maxsize)) {
                    if (!strlen($child->node_value())) {
                        $child->unlink_node();
                        break;
                    }
                    $text = $this->dom->create_text_node(html_entity_decode(substr($data, strlen($words)),
    $this->quotes));
                    $child->replace_node($text);
                    $buf .= $words;
                    break;
                } else { // situacija, kai netelpa nei vienas zodis arba jam patalpinti nera vietos
                    if (!($pos = strpos($data, ' '))) {
                        // $this->fallback = true;
                        $buf .= $this->dom->dump_node($child);
                        // $buf .= $child->node_value();
                        $child->unlink_node();
                    } else {
                        $cut = substr($data, 0, $pos + 1);
                        $text = $this->dom->create_text_node(html_entity_decode(substr($data, strlen($cut)),
    $this->quotes));
                        $child->replace_node($text);
                        $buf .= $cut;
                    }
                    break;
                }
            } else {
                $this->nextPage = true;
                break;
            }
        }
    }
}

```

```

    }
}

if ($this->page == 0) $this->nextSkip = $this->len($buf);

}

if (strlen($buf) && !$this->nextPage) $this->nextPage = true;

return $buf;
}

function getNodeSize($node) {
    if (is_null($node)) return 0;
    $size = 0;
    $mysize = 0;
    if ($node->node_type() == XML_ELEMENT_NODE) {
        $mysize += $this->len($node->tagname()) + 2;
        if ($attrArray = $node->attributes()) {
            // parse attributes
            foreach($attrArray AS $attr) {
                $mysize += $this->len($attr->name()) + $this->len(htmlspecialchars($attr->value())) + 4;
            }
        }
        if ($schildArray = $node->child_nodes()) {
            $mysize += $this->len($node->tagname()) + 3;
            // add child nodes
            foreach($schildArray AS $schild) {
                $size += $this->getNodeSize($schild);
            }
        } else {
            $mysize += 1;
        }
    }

    if (!in_array($node->tagname(), $this->systemtags)) $size += $mysize;

} else {
    // this is a CONTENT NODE
    // $size += $this->len($node->node_value());
    $size += $this->len(htmlspecialchars($node->node_value(), $this->quotes));
}

return $size;
}

```